Approximation algorithms to the capacitated tree-routings in networks(Dissertation_全文)

AUTHOR(S):

Ehab Ibrahim, Ibrahim Morsy

Approximation Algorithms to the Capacitated Tree-Routings in Networks

By

Ehab Ibrahim Ibrahim Morsy

Doctoral Dissertation
Submitted to Graduate School of Informatics,
Kyoto University in Partial Fulfillment of
the Requirements for the Degree
of
DOCTOR OF INFORMATICS

Kyoto University
September, 2009

Preface

Routing is a process of selecting paths in a network along which traffics between two groups of sources and destinations are sent. This process is a fundamental task in several network designs such as the internet, telecommunication, and transportation networks. The implementation of routing process in networks may induce several difficult combinatorial optimization problems. In this thesis, we study several routing protocols which are translated into combinatorial optimization problems.

An optimization problem consists in finding the best solution among a large set of feasible solutions to the underlying problem, where the evaluation of a solution is determined by the objective function of the problem. A solution that optimizes the objective function is called an optimal (exact) solution. It is well known that the most combinatorial optimization problems involved in real applications are difficult to be solved exactly, i.e., they are NP-hard problems. This implies that constructing exact solutions to these problems would be prohibitively time consuming since it is believed that an NP-hard problem cannot be efficiently solved in polynomial time of the input size. However, most practical applications ask for a solution sufficiently close to the optimal. In this sense, the design of efficient approximation algorithms has a major attention in the last years. An approximation algorithm usually computes in polynomial time of the input size a solution for which the value of the objective function is close to the optimal value.

In this thesis, we study several models of the capacitated routing problem in edge-weighted networks. For each of them, we are given a set of vertices each of which has a nonnegative demand, a set of vertices each of which has a nonnegative opening cost as a sink (or a source), and some capacity constraints, and we are asked to construct a specific routing structure of minimum cost. All problems studied in this thesis are NP-hard and hence our aim is to propose a polynomial time approximation algorithm for each of them under capacity constraints.

First, we study the capacitated multicast routing problem under multi-tree model, in which we are interested in constructing a minimum cost set of trees on all vertices with

nonzero demands each of which is rooted at a prescribed vertex (called source) and has a limited amount of demand by a demand capacity constraint. We also extend this model to the case where more than one vertex are nominated to be sources with an extra cost.

Next, we consider a special case of the well-known single-sink buy-at-bulk problem, in which we are given one cable type, and we wish to construct a set of paths of minimum cost along which demands of all vertices are sent to a single sink such that the demand of each vertex is sent through a single path, that is, it is not allowed to split the demand of any vertex.

Finally, we present a more general formulation of the capacitated routing problem which includes several well-known routing problems as its special cases. It also includes some fundamental problems such as Steiner tree problem and bin packing problem. In particular, the problem consists of finding a minimum cost set of tree-routings under specific demand and edge capacity constraints.

Our approximation algorithms designed for the above capacitated routing problems are based on a variety of new results on tree covers and tree partitions.

We believe that our algorithms for the above problems are useful as the theoretical foundation to practical algorithms and developed techniques give a new insight into the theoretical structure of capacitated routing problems. We hope that the works in this thesis will be helpful to advance the study in these topics.

September, 2009

Ehab Morsy

Acknowledgment

Without support and encouragement from numerous people, I could have never completed this work.

First of all, I extend my sincere thanks to **The Egyptian Ministry of Higher Education** for managing the scholarship program and financing my entire study in Japan for four years and to the Department of Mathematics of the Faculty of Science, Suez Canal University for nominating me for this scholarship.

I would like to express my heartily grateful and sincere appreciation to Professor **Hiroshi Nagamochi** of Kyoto University for his enthusiastic guidance and persistent encouragement. He gave me continuous support especially when I start studying in this area. His advice provided insights into research activity for me. Enthusiastic discussion with him were quite exciting and invaluable experience for me. Without his considerable help, none of this work could have been completed.

I wish to express my gratitude to Professor Professor **Liang Zhao** of Kyoto University, Professor **Takuro Fukunaga** of Kyoto University, and all members in Discrete Mathematics laboratory of Kyoto University who offered me a friendly environment for my study besides their help in my personal life.

I am also thankful to Professor **Masao Fukushima** of Kyoto University and Professor **Yoshito Ohta** of Kyoto University for serving on my dissertation committee.

I would like to express my heartiest gratitude to my parents, my brothers and sisters, my wife, and my kids for their heartfelt support and encouragement.

Above all, my deep gratefulness is to **ALLAH** who gives me health that enables me to have strength, courage and determination to finish this work properly.

Contents

1 Introduction **1**

1.1 Notations 1
1.2 Approximation algorithms 2
1.3 Steiner tree problem 3
1.4 Balancing minimum Steiner and shortest path trees 4
1.5 Uncapacitated facility location problem 7
1.6 Single-sink buy-at-bulk problem 8
1.7 Organization of the thesis 9

2 The Capacitated Multicast Routing Problem in Networks **13**

2.1 Introduction 13
2.2 General Demands 17
2.3 Tree Cover 19
2.3.1 Tree covers in special cases 19
2.3.2 Algorithm for tree cover 24
2.4 Approximation algorithm 29

3 Multicast Routing Problem in a Network with Multi-sources **33**

3.1 Introduction 33
3.2 Preliminaries 35
3.3 General demands 36
3.4 Unit demands 40

4 The Minimum Cost Edge Installation Problem for Routings **45**

4.1 Introduction 45
4.2 Preliminaries 47
4.3 Tree partition 48
4.3.1 Tree partition in special trees 48
4.3.2 Algorithm for tree partition 54
4.4 Approximation algorithm to MCEI 57

5 The Capacitated Tree-Routing Problem in Networks **61**
5.1 Introduction 61
5.2 Simple approximation algorithm 63
5.3 Improved approximation algorithm 65
5.4 Proof of Lemma 5.3 68

6 The Generalized Capacitated Tree-routing Problem **73**
6.1 Introduction 73
6.2 Preliminaries 76
6.3 Approximation algorithm for $\lambda \geq \alpha + \beta\kappa$ 78
6.3.1 Approximation algorithm for $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor = 1$ 78
6.3.2 Approximation algorithm for $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor \geq 2$ 80
6.4 Approximation algorithm for $\lambda < \alpha + \beta\kappa$ 82
6.4.1 Generalized capacitated network design problem 82
6.4.2 Approximation algorithms to GCTR 85
6.5 Approximation algorithm to FGCTR 89

7 Conclusion **91**

List of Figures

1.1 Illustration for the Steiner tree problem. 3

1.2 Illustration for balanced trees; (a) a given graph G; (b) a minimum spanning tree of G; (c) a shortest path tree in G rooted at s; (d) a $(2, 11/9)$-balanced tree in G. 5

1.3 Illustration for UFL. 7

1.4 Illustration for the problems studied in this thesis and possible future work. . 12

2.1 Illustration for CMTR; (a) an instance of CMTR; (b) a feasible solution to the instance in (a); (c) an optimal solution to the instance in (a). 14

2.2 Illustration for Lemma 2.4; (a) a tree T_x defined in Lemma 2.4; (b) a tree T'_v obtained from T_x by duplicating P. 21

2.3 Illustration for Lemma 2.5; (a) the construction of the tree T_x defined in Lemma 2.5; (b) illustration for Case 1 in Lemma 2.5; (c) illustration for Case 2 in Lemma 2.5. 22

2.4 Illustration of Lemma 2.6; (a) a branch B of T_x such that T_u is a 2/3-balance-tree and $\kappa < |Z_x \cap V(B)| < (4/3)\kappa$; (b) $w(B') \leq \min\{w(B_1), w(B_2)\}$; (c) $w(B_1) \leq \min\{w(B_2), w(B')\}$. 23

2.5 Illustration of one iteration of the while-loop in algorithm TreeCover. . . . 26

2.6 Illustration of algorithm TreeCover; (a) a minimum tree of the graph given in Fig. 2.1(a); (b) and (c) illustrate intermediate iterations of algorithm TreeCover applying to the tree in (a). 27

3.1 Illustration for algorithm GeneralCMMTR applying to an instance of CMMTR with source set $S = \{s_1, \ldots, s_6\}$ and terminal set $M = \{u_1, \ldots, u_{13}\}$; (a) a ρ_{UFL}-approximate solution to UFL instance I' defined in Steps 1; (b) a ρ_{ST}-approximate solution to the Steiner tree problem to $(G', w, M \cup \{r\})$ defined in Step 2. 37

3.2 Output of algorithm GeneralCMMTR applied to example given in Fig. 3.1. 38

4.1 (a) illustration of Case 2 in Lemma 4.3, where L and N are represented by white and gray regions, respectively; (b) illustration of a tree T_x in Lemma 4.4; (c) a tree T' obtained by rerooting T_x at vertex v. 49
4.2 Illustration of Case 2 in Lemma 4.5, where A, B, and C are represented by white, gray, and black regions, respectively; (a) $t_C \in Z_v^2$; (b) $t_C \in Z_u^1$. 51
4.3 Illustration of Case 3 in Lemma 4.5, where A, B, and C are represented by white, gray, and black regions, respectively. 53

5.1 Illustration of swapping process, where Z_4 and Z_5' are swapped between $\mathcal{M}_{dwn}(e)$ and $\mathcal{M}_{up}(e)$; (a) $\mathcal{M}_{dwn}(e) = \{Z_1, Z_2, Z_3, Z_4\}$ and $\mathcal{M}_{up}(e) = \{Z_4', Z_5'\}$; (b) $\mathcal{M}_{dwn}(e) = \{Z_1, Z_2, Z_3\}$ and $\mathcal{M}_{up}(e) = \{Z_4'\}$. 66
5.2 (a) the construction of a collection $\mathcal{C}_j$ (gray triangles) computed in line 6 of algorithm TreeCover, where t_j is the terminal of the minimum distance d in T_v; (b) illustration for Lemma 5.3(iii)(b); (c) illustration for Lemma 5.3(iii)(c). 69

List of Tables

1.1 Approximation algorithms for the Steiner tree problem. 4
1.2 Approximation algorithms for UFL. 6
1.3 Approximation algorithms for SSBB and DSSBB. 9

3.1 Approximation algorithms for CCFL and CMMTR problems 34

6.1 Approximation algorithms for CND, CMTR, CTR, and GCTR problems, where $\theta = \lceil \lambda/(\alpha + \beta\kappa) \rceil / \lfloor \lambda/(\alpha + \beta\kappa) \rfloor$. 77

Chapter 1

Introduction

In this chapter, we first describe notations and definitions which will be used in the rest of the thesis. Next we discuss some fundamental combinatorial optimization problems such as Steiner tree problem, balancing minimum Steiner and shortest paths trees, and the uncapacitated facility location problem. These problems will be used in designing our algorithms in the subsequent chapters. Finally, we give an overview of the organization of the thesis.

1.1 Notations

This section introduces some notations and definitions.

Let Z^+ and R^+ denote the sets of nonnegative integers and nonnegative reals, respectively.

For a set Z, a set $\{Z_1, Z_2, \ldots, Z_\ell\}$ of pairwise disjoint non-empty subsets of Z is called a *partition* of Z if $\cup_{i=1}^{\ell} Z_i = Z$.

Let G be a simple undirected graph. We denote by $V(G)$ and $E(G)$ the sets of vertices and edges in G, respectively. For a subgraph G' of a graph G, let $G - G'$ denote the subgraph induced from G by $E(G) - E(G')$. Similarly, for two subgraphs G' and G'' of graph G, let $G' + G''$ denote the subgraph induced from G by $E(G') \cup E(G'')$. For an edge-weighted graph (G, w) with a nonnegative weight function $w : E(G) \to R^+$, the length of a shortest path between two vertices u and v is denoted by $d_{(G,w)}(u, v)$. We may use $w(u, v)$ to denote the weight of an edge $(u, v) \in E(G)$. An edge-weighted graph (G, w) is called *metric* if the triangle inequality holds, i.e., $w(x, z) \leq w(x, y) + w(y, z)$ for every $x, y, z \in V(G)$. For a subgraph H of G, let $w(H)$ denote the sum of weights of all edges in H. Given a demand function $q : V(G) \to R^+$ and a subgraph H of G, we use $q(H)$ and $q(V(H))$ interchangeably to denote the sum $\sum_{v \in V(H)} q(v)$ of demands of all vertices in $V(H)$.

Let T be a tree. A *subtree* of T is a connected subgraph of the tree. A set of subtrees in T is called a *tree cover* if each vertex in T is contained in at least one of the subtrees. For a subset $X \subseteq V(T)$ of vertices, let $T\langle X \rangle$ denote the minimal subtree of T that contains X. Note that $T\langle X \rangle$ is uniquely determined.

Now we regard T as a rooted tree. Let $L(T)$ denote the set of leaves in T. For a vertex v in T, let $Ch(v)$ and $D(v)$ denote the sets of children and descendants of v, respectively, where $D(v)$ includes v. A *subtree T_v rooted at v* is the subtree induced by $D(v)$, i.e., $T_v = T\langle D(v)\rangle$. For an edge $e = (u, v)$ in a rooted tree T, where $u \in Ch(v)$, the subtree induced by $D(u) \cup \{v\}$ is denoted by T_e, and is called a *branch* of T_v. For a rooted tree T_v, the *depth* of a vertex u in T_v is the length (the number of edges) of the path from v to u.

Let G be a connected graph. A spanning tree of G is a tree that connects all the vertices in $V(G)$ together. Given a nonnegative weight $w(e)$ for each edge $e \in E(G)$, a *minimum spanning tree* of G is a tree of the minimum total weight among all spanning trees of G. A *shortest path tree* T of (G, w) is a spanning tree of G constructed so that the distance between a selected root vertex $s \in V(G)$ and all other vertices is minimal, i.e., $d_{(T,w)}(s, v) = d_{(G,w)}(s, v)$ for all $v \in V(G)$.

Throughout the thesis we study different routing models in networks with nonnegative demand function on its vertices, and we wish to route the demands of all vertices to a specified set of vertices in the network (called sinks or sources). The "flow" on each edge of the network refers to the total demand that goes along the edge when the demands of all vertices are routed to the specified sinks or sources simultaneously.

1.2 Approximation algorithms

Many combinatorial optimization problems have been recognized as NP-hard problems. First, Cook [14] proved that SAT is NP-complete problem. In the subsequent years, the foundations of the theory of NP-completeness were established [22]. Since then many combinatorial optimization problems are proved to be NP-hard, see for example [3, 18, 22, 34]. Most people believe that NP-hard problems cannot be solved exactly in polynomial time. An algorithm for a given problem is said to be a polynomial-time algorithm if its running time is $O(n^c)$, where n denotes the input size of a problem instance and c is a constant.

One option of dealing with NP-hard problems is to investigate approximation algorithms; an algorithm that runs in polynomial time of the input size and returns a solution of cost close to the optimal value. Namely, for a minimization problem, approximation algorithms compute a feasible solution to the problem such that:

(i) The algorithm terminates after performing its steps which number is bounded from above by a polynomial in the input size of a given instance, and

(ii) The cost of the obtained solution of a given instance is bounded from above by α times the value of an optimal solution of the instance.

Approximation algorithms for the maximization problems are defined similarly except for the obtained solution is bounded from below by $1/\alpha$ times the value of any optimal solution. α is called the *approximation ratio*, the *performance ratio*, or the *approximation guarantee* of

the algorithm. Obviously, α is greater than or equal 1 for the minimization and maximization problems. See [2] for a survey of definitions and developments of approximation algorithms.

Note that the theory of NP-completeness can provide an evidence not only that it is hard to solve a problem precisely but also that it is hard to obtain an approximate solution to a problem within a certain accuracy.

1.3 Steiner tree problem

The purpose of this section is to provide some known results about the Steiner tree problem which will be a basic tool in approximation algorithms given in the subsequent chapters.

Given a connected graph $G = (V, E)$ with edge weights $w(e) \geq 0$, $e \in E$, and a prescribed subset $Z \subseteq V$ of terminals, the *Steiner tree problem* asks to find a minimum weighted tree T of G with $Z \subseteq V(T)$. The vertices in $V(T) - Z$ are called Steiner vertices of T. The Steiner tree problem is a classical NP-hard optimization problem even with Euclidean or rectilinear costs [21]. The problem remains NP-hard even for unweighted graphs, i.e., all edges of the graph have unit weights [42]. So it is unlikely to find a polynomial-time algorithm that returns exact solution to the problem unless P=NP. This motivates designing efficient approximation algorithms (polynomial-time algorithms) to the problem that return an approximate solution whose cost is not far from the optimal value (the cost of a minimum Steiner tree).

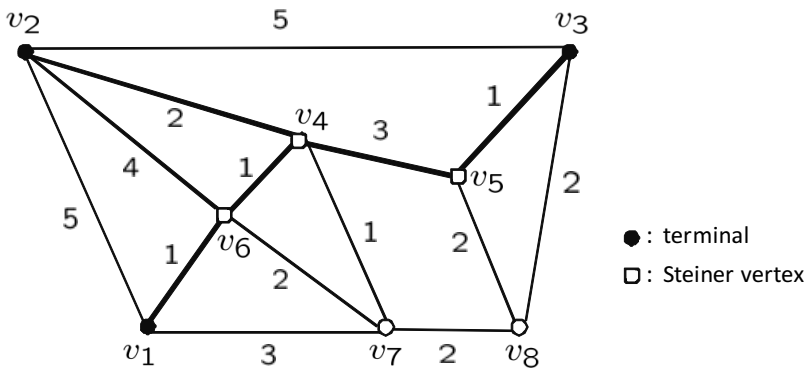

Figure 1.1: Illustration for the Steiner tree problem.

Figure 1.1 illustrates the Steiner tree problem, where $V = \{v_1, v_2, \ldots, v_8\}$, $Z = \{v_1, v_2, v_3\}$ is the set of terminals, and the number beside each edge represents its weight. For this example, the minimum Steiner tree is induced by the thick edges and $\{v_4, v_5, v_6\}$ is the set of Steiner vertices of this tree.

Based on the minimum spanning tree, the first approximation algorithm for the Steiner tree problem is mentioned by Moore [23]. This algorithm achieves an approximation ratio of 2. Note that the minimum spanning tree problem is equivalent to the Steiner tree problem in the case where all vertices in the graph are terminals. It is well known that the minimum spanning

Table 1.1: Approximation algorithms for the Steiner tree problem.

Reference	**Approx. Ratio**
Moore, see [23]	2
Zelikovsky [71]	1.834
Berman and Ramaiyer [8]	1.734
Zelikovsky [72]	1.694
Prömmel and Steger [59]	1.667
Karpinsky and Zelikovsky [43]	1.644
Hougardy and Prömmel [35]	1.598
Robins and Zelikovsky [62]	1.55

tree of a graph can be computed in polynomial time, see for example algorithms in [46, 58]. This ratio remains the best known for more than 20 years until Zelikovsky [71] proposed the idea of k-Steiner tree for the analysis of approximation algorithms. Based on the new analysis, Zelikovsky [71] gave a 11/6-approximation algorithm to the Steiner tree problem. All the current known approximation algorithms for the Steiner tree problem use the idea of k-Steiner tree in their analysis. Berman and Ramaiyer [8] proposed a family of algorithms which achieves a performance ratio of 1.734 for sufficiently large k in k-Steiner trees. By using a new kind of analysis, Zelikovsky [72] gave a relative greedy 1.694-approximation algorithm. The main idea of such a relative greedy algorithm is to start with a Steiner tree that is obtained via the minimum spanning tree in a complete graph with vertex set Z and the weight of each edge in the graph equals the length of a shortest path between end terminals of the edge in G. This solution is repeatedly improved by adding certain minimum Steiner trees on at most k terminals and deleting the resulting cycles. Note that for a constant number of terminals a minimum Steiner tree can be computed in polynomial time [19]. Karpinsky and Zelikovsky [43] used an idea based on the concept of *loss* of a Steiner tree to derive a 1.644 approximation ratio. Afterwards Hougardy and Prömmel [35] generalized their idea to prove an approximation ratio of 1.598. Recently, Robins and Zelikovsky [62] incorporated the idea of the loss of Steiner tree into a relative greedy algorithm to obtain a Steiner tree of cost within $1 + \frac{\ln 3}{2} < 1.55$ of the cost of the minimum Steiner tree. Up to this moment, this is the best known approximation factor for the Steiner tree problem. Robins and Zelikovsky [62] also showed that this factor is reduced to about 1.28 for quasi-bipartite graphs.

Table 1.1 summarizes proposed approximation ratios to the Steiner tree problem.

1.4 Balancing minimum Steiner and shortest path trees

Let $G = (V, E)$ be a connected graph with edge weights $w(e) \geq 0$, $e \in E$, and a set $Z \subseteq V$ of terminals. Most network design problems arising in practical applications ask for computing

a minimum Steiner tree that spans the set Z of all terminals. If all vertices in G are terminals, then the problem becomes the *minimum spanning tree problem*. On the other hand, we may wish to send messages from a designated vertex $s \in V$ (the root) to all terminals in the same network. In this case, messages may be required to be sent to the terminals along short paths so that the messages reach their destinations quickly. For example, in the VLSI design, interconnect delay has become an important factor, and minimum interconnect delay is achieved when the spanning tree is a shortest path tree rooted at s.

It is possible for some instances that the cost of a shortest path tree is more significant than that of a minimum spanning tree [44]. Similarly, the ratio of the distance between the root and the furthest terminal in all minimum spanning trees over the shortest distance between them may also be unboundedly large. Namely, it can be as large as $O(n)$ [9, 44], where n denotes the number of vertices in the graph.

As both of the cost of a spanning tree and the distance from the root to each terminal are important factors in practical applications, the attention is turned to find a spanning tree whose total cost is not much more the cost of the minimum Steiner tree and the distance from the root to each terminal is not much more than the distance in the shortest path tree. Namely, given a minimum Steiner tree and a shortest path tree on $(G, w, Z \cup \{s\})$, a "balanced" Steiner tree T is a Steiner tree of G that spans $Z \cup \{s\}$ and approximates both the shortest path tree and the minimum Steiner tree. That is, there are constants $\alpha, \beta \geq 1$ such that

(i) the distance between s and any vertex $v \in Z$ in T is at most α times the shortest distance between s and v in G, and

(ii) the cost of T is at most β times the cost of a minimum Steiner tree.

We say that an algorithm that computes such a balanced tree T has an approximation factor of (α, β).

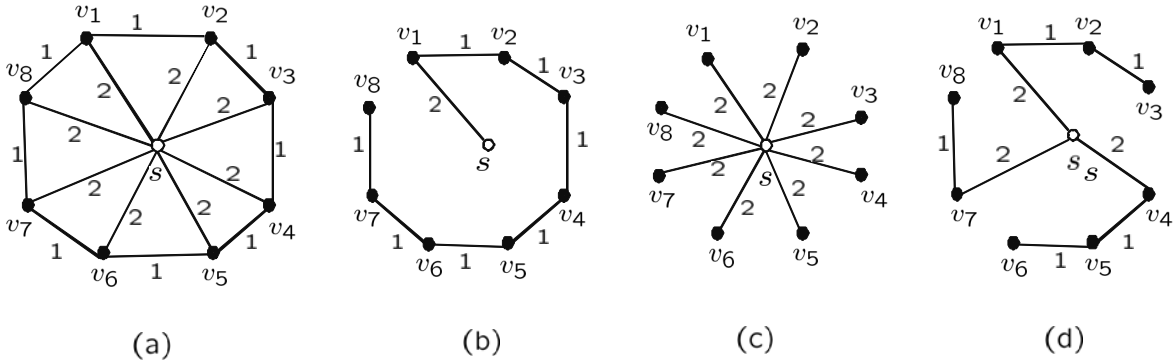

Figure 1.2: Illustration for balanced trees; (a) a given graph G; (b) a minimum spanning tree of G; (c) a shortest path tree in G rooted at s; (d) a $(2, 11/9)$-balanced tree in G.

Consider the graph G described in Fig. 1.2(a), where $V(G) = \{s, v_1, v_2, \ldots, v_8\}$ and the number beside each edge represents its weight. Any minimum spanning tree of G has a total weight of 9 and consists of exactly one edge incident to s and 7 unit weight edges (see Fig. 1.2(b)). On the other hand, there is a unique shortest path tree of G rooted at s that consists of only all edges incident to s (see Fig. 1.2(c)). The cost of the shortest path between s and any vertex in $V(G) - \{s\}$ is 2. In Fig. 1.2(d), a $(2, 11/9)$-balanced tree T in G are described since the weight of T is 11 and the weight of the path between s and the furthest vertex in $V(G) - \{s\}$ is 4.

See [16, 17] for applications of a balanced tree to VLSI.

For a balanced tree approximating the minimum spanning tree and the shortest path tree, Awerbuch et al. [5] gave a $(\alpha, 1 + 4/(\alpha - 1))$-approximation algorithm in $O(m + n \log n)$ time, where m denotes the number of edges in the underlying graph and $\alpha > 1$ is a prescribed constant. Afterwards, Khuller et al. [44] proposed a $(\alpha, 1 + 2/(\alpha - 1))$-approximation algorithm. Given the minimum spanning tree and the shortest path tree, the algorithm of Khuller et al. [44] runs in linear time in the number of vertices. It is not difficult to deduce an algorithm with performance factor $(\alpha, \rho_{\mathrm{ST}}(1 + 2/(\alpha - 1)))$ that approximate the minimum Steiner tree and the shortest path tree [50], where ρ_{ST} is any approximation factor achievable for the Steiner tree problem.

Table 1.2: Approximation algorithms for UFL.

Reference	**Approx. Ratio**	**Technique**
Shmoys et al. [66]	3.16	LP rounding
Guha and Khuller [27]	2.47	LP rounding+ greedy augmentation
Chudak [13]	1.736	LP rounding
Korupolu et al. [45]	5+ϵ	Local search
Jain and Vazirani [38]	3	Primal-dual
Charikar and Guha [12]	1.853	Primal-dual+ greedy augmentation
Charikar and Guha [12]	1.728	LP rounding+ Primal-dual+ greedy augmentation
Mahdian et al. [51]	1.861	Greedy algorithm
Jain et al. [37]	1.61	Greedy algorithm
Sviridenko [67]	1.582	LP rounding
Mahdian et al. [52]	1.52	Greedy algorithm+ greedy augmentation

1.5 Uncapacitated facility location problem

The facility location problem is the problem of locating facilities to effectively serve a set of clients. Several variants of this problem have been studied extensively in the operation research literatures and have received considerable attention in the area of approximation algorithms. In this section we discuss the basic facility location problem, *the uncapacitated facility location problem* (UFL). In addition to the wide area of practical applications in which UFL is involved, its importance may also includes dealing with more complicated location models. UFL will be used in approximating a multicast tree routing problem discussed in Chapter 3.

UFL is formulated as follows. An instance (G, c, F, f, C, b) of UFL consists of an undirected graph G, an edge weight function $c : E(G) \to R^+$, a set F of facilities, an opening cost function $f : F \to R^+$, a set $C = V(G) - F$ of clients, and a demand function $b : C \to R^+$, where $f(i)$ means the cost of opening facility i, and $c(i, j)$ means the cost for connecting facility $i \in F$ and client $j \in C$. The goal is to identify a subset $F' \subseteq F$ of facilities to open that minimizes the opening cost and the connecting cost, i.e.,

$$\Phi(F') := \sum_{i \in F'} f(i) + \sum_{j \in C} b(j) \min_{i \in F'} c(i, j).$$

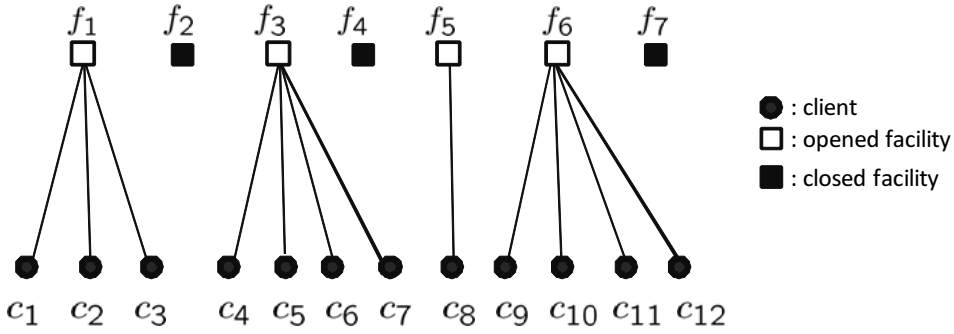

Figure 1.3: Illustration for UFL.

Figure 1.3 provides an example of UFL, where $F = \{f_1, \ldots, f_7\}$ and $C = \{c_1, \ldots, c_{12}\}$ are the sets of facilities and clients, respectively. A subset $F' = \{f_1, f_3, f_5, f_6\}$ of facilities are opened such that $\{c_1, c_2, c_3\}$, $\{c_4, c_5, c_6, c_7\}$, $\{c_8\}$, $\{c_9, c_{10}, c_{11}, c_{12}\}$ are served by f_1, f_3, f_5, and f_6, respectively.

This problem has many applications in operation research [15, 47], network design problems such as placements of routers and caches [28, 45], agglomeration of traffic or data [1, 29], and web server replications in a content distribution network [39, 60]

UFL is NP-hard even if (G, c) is metric. various approaches have been proposed for UFL such as LP rounding, primal-dual method, local search, and a combination of these methods. The first constant factor approximation algorithm is given by Shymoys et al. [66]. Since then

a large number of approximation algorithms have been proposed for UFL [12, 13, 27, 37, 38, 45, 51, 67].

Recently, Mahdian et al. [52] combined the greedy algorithm of [37] and the greedy augmentation of [12, 27] to propose the current best approximation ratio 1.52 for UFL. Guha and Khuller [27] proved that it is impossible to get an approximation factor of 1.463 for the UFL unless NP$\subseteq$DTIME$[n^{O(\log \log n)}]$.

Table 1.2 summarizes a series of approximation algorithms proposed to UFL.

1.6 Single-sink buy-at-bulk problem

Consider a connected graph $G = (V, E)$ with edge weights $w(e) \geq 0$, $e \in E$. We are given a set $M \subseteq V$ of vertices specified as sources and a vertex $s \in V$ specified as a sink. Each source $v \in M$ has a nonnegative demand $q(v)$, all of which must be sent to s through a single path. We are also given a finite set of different cable types, where each cable type i has capacity u_i and cost c_i (per unit weight). That is, installing a copy of cable type i on an edge e costs $c_i w(e)$. The value c_i/u_i refers to the cost per unit capacity per unit weight of cable type i. Note that if $u_i \leq u_j$ and $c_i \geq c_j$, then we can eliminate cable type i from consideration. Thus we can assume without loss of generality that the cables are ordered such that $u_i < u_j$ and $c_i < c_j$ for all $i < j$. Moreover, it holds $c_j/u_j < c_i/u_i$ for each $i < j$ since otherwise cable type j can be replaced by u_j/u_i copies of cable type i without increase the cost. In other words, the costs of cables obey economies of scale, i.e., the cost per unit capacity per unit weight of a high capacity cable is significantly less than that of a low capacity cable. The *single-sink buy-at-bulk problem* (SSBB) (also known as the *single-sink edge installation problem* [30]) asks to construct a network of cables in the graph by installing an integer number of each cable type between adjacent vertices in G so that the given demands at the sources can be routed simultaneously to sink s. The goal is to minimize the costs of installed cables. When a demand of each source v is allowed to be routed to the sink along multiple paths (i.e., $q(v)$ is splittable), the problem is called the *divisible single-sink buy-at-bulk problem* (DSSBB) [40].

SSBB has applications in design of telecommunication networks. Also, DSSBB are involved in practical applications such as routing oil from several oil wells to a major refinery.

The problem of buy-at-bulk network design was first introduced by Salman et al. [65]. They proved that the problem is NP-hard by showing a reduction from the Steiner tree problem. Moreover, they showed that the problem remains NP-hard even when only one cable type is available. They also gave an $O(\log n)$-approximation algorithm for SSBB in the Euclidean space, where n denotes the number of vertices in the graph. Awerbuch and Azar [4] gave an $O(\log^2 n)$-approximation algorithm for SSBB in the general metric space. Based on LP rounding, Garg et al. [20] presented an $O(K)$-approximation algorithm, where K denotes the number of cable types. Afterwards Tawlar [68] proved that the algorithm given by Guha et al. [30] has an approximation ratio of about 2000. He also proved a 216 approximation

Table 1.3: Approximation algorithms for SSBB and DSSBB.

Reference	**Approx. Ratio**	**SSBB/DSSBB**
Awerburch and Azar [4]	$O(\log^2 n)$	SSBB
Salman et al. [65]	$O(\log n)$	SSBB in R^d
Garg et al. [20]	$O(K)$	SSBB
Guha et al. [30, 68]	2000	SSBB
Talwar [68]	216	SSBB
Jothi and Raghavachari [41]	145.6	SSBB
Gupta et al. [31]	72.8	DSSBB
Jothi and Raghavachari [41]	65.49	DSSBB
Grandoni and Italiano [24]	24.92	DSSBB

ratio to the problem. Recently, Jothi and Raghavachari [41] proposed a 145.6-approximation algorithm for SSBB.

Obviously, algorithms designed for SSBB have the same ratio when applied to DSSBB instances. In addition, DSSBB itself has received attentions in the recent study. Meyerson et al. [54] proved a $O(\log n)$-approximation ratio. Gupta et al. [31] gave a 72.8-approximation algorithm. This ratio is reduced by Jothi and Raghavachari [41] to 65.49. Recently, Grandoni and Italiano [24] presented a 24.92-approximation algorithm to DSSBB.

Table 1.3 summarizes approximation ratios known for SSBB and DSSBB.

1.7 Organization of the thesis

In addition to the last chapter which concludes the thesis, the rest of this thesis is structured as follows.

Chapter 2: The Capacitated Multicast Routing Problem in Networks

Let $G = (V, E)$ be a connected graph such that each edge $e \in E$ is weighted by nonnegative real $w(e)$. Let κ be a positive real, $s \in V$ be a vertex designated as a source, and $M \subseteq V - \{s\}$ be a set of terminals with nonnegative demands $q(v)$, $v \in M$. The *capacitated multicast tree routing problem* (CMTR) asks to find a partition $\{Z_1, Z_2, \ldots, Z_\ell\}$ of M and a set $\{T_1, T_2, \ldots, T_\ell\}$ of trees of G such that, for each i, the total demand in Z_i is at most κ and each T_i spans $Z_i \cup \{s\}$. The objective is to minimize $\sum_{i=1}^{\ell} w(T_i)$. We propose a $(2 + \rho_{\mathrm{ST}})$-approximation algorithm to CMTR with general demand and a $(3/2 + (4/3)\rho_{\mathrm{ST}})$-approximation algorithm to CMTR with unit demand, where ρ_{ST} is any achievable approximation ratio for the Steiner tree problem. Our algorithms are based on elaborate tree covers of a given tree.

Chapter 3: Multicast Routing Problem in a Network with Multi-sources

We consider the *capacitated multi-source multicast tree routing problem* (CMMTR) in an undirected graph $G = (V, E)$ with an edge weight $w(e) \geq 0$, $e \in E$. We are given a real number $\kappa > 0$, a source set $S \subseteq V$ with a weight $g(e) \geq 0$, $e \in S$, and a terminal set $M \subseteq V - S$ with a demand function $q : M \to R^+$, where $g(s)$ means the cost for opening a vertex $s \in S$ as a source in a multicast tree. Then CMMTR asks to find a subset $S' \subseteq S$, a partition $\{Z_1, Z_2, \ldots, Z_\ell\}$ of M, and a set $\{T_1, T_2, \ldots, T_\ell\}$ of trees of G such that, for each i, the total demand in $q(Z_i)$ is at most κ and T_i spans $Z_i \cup \{s\}$ for some $s \in S'$. The objective is to minimize the sum of the opening cost of S' and the constructing cost of $\{T_i\}$, i.e., $\sum_{s \in S'} g(s) + \sum_{i=1}^{\ell} w(T_i)$. We propose a $(2\rho_{\text{UFL}} + \rho_{\text{ST}})$-approximation algorithm to the CMMTR, where ρ_{UFL} is any approximation ratio achievable for UFL. When all terminals have unit demands, we give a $((3/2)\rho_{\text{UFL}} + (4/3)\rho_{\text{ST}})$-approximation algorithm.

Chapter 4: The Minimum Cost Edge Installation Problem for Routings

We consider the *minimum cost edge installation problem* (MCEI) in a graph $G = (V, E)$ with edge weight $w(e) \geq 0$, $e \in E$. We are given an edge capacity $\lambda > 0$, a vertex $s \in V$ designated as a sink, and a source set $M \subseteq V - \{s\}$ with demand $q(v) \in [0, \lambda]$, $v \in M$. For any edge $e \in E$, we are allowed to install an integer number $h(e)$ of copies of e. MCEI asks to send demand $q(v)$ from each source $v \in M$ along a single path P_v to the sink s, but not allowed to split the demand of any $v \in M$. For each edge $e \in E$, a set of such paths can pass through a single copy of e in G as long as the total demand along the paths does not exceed the edge capacity λ. The objective is to find a set $\mathcal{P} = \{P_v \mid v \in M\}$ of paths of G that minimizes the installing cost $\sum_{e \in E} h(e)w(e)$. We propose a $(15/8 + \rho_{\text{ST}})$-approximation algorithm to MCEI.

Chapter 5: The Capacitated Tree-Routing Problem in Networks

Let $G = (V, E)$ be a connected graph such that each edge $e \in E$ is weighted by a nonnegative real $w(e)$. Let $\kappa > 0$ be a routing capacity, $\lambda \geq 1$ be an integer edge capacity, s be a vertex designated as a sink, and $M \subseteq V - \{s\}$ be a set of terminals with a demand function $q : M \to R^+$. The *capacitated tree-routing problem* (CTR) asks to find a partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_\ell\}$ of M and a set $\mathcal{T} = \{T_1, T_2, \ldots, T_\ell\}$ of trees of G such that, for each i, the total demand in Z_i is at most κ and T_i spans $Z_i \cup \{s\}$. A single copy of an edge $e \in E$ can be shared by at most λ trees in $\mathcal{T}$; any integer number of copies of e are allowed to be installed, where the cost of installing a copy of e is $w(e)$. The objective is to find a solution $(\mathcal{M}, \mathcal{T})$ that minimizes the total installing cost. This new routing problem formulation unifies several important routing problems such as the *capacitated network design* (CND) and

CMTR problems. We propose a $(2+\rho_{\mathrm{ST}})$-approximation algorithm to CTR.

Chapter 6: The Generalized Capacitated Tree-routing Problem

In this chapter, we study the *generalized capacitated tree-routing problem* (GCTR), which was introduced to unify several known multicast problems in networks with edge/demand capacities. Let $G=(V,E)$ be a connected graph with an edge weight $w(e)\geq 0$, $e\in E$, and a bulk edge capacity $\lambda>0$; we are allowed to construct a network on G by installing any edge capacity $h(e)\lambda$ with an integer $h(e)\geq 0$ for each edge $e\in E$, where the resulting network costs $\sum_{e\in E}h(e)w(e)$. Given a demand capacity $\kappa>0$, prescribed constants $\alpha,\beta\geq 0$, a sink $s\in V$, and a set $M\subseteq V$ of terminals with a demand $q(v)\geq 0$, $v\in M$, we wish to construct the minimum cost network so that all the demands can be sent to s along a suitable collection $\mathcal{T}=\{T_1,T_2,\ldots,T_\ell\}$ of trees rooted at s, where the total demand collected by each tree T_i is bounded from above by κ, and the flow amount $f(e)$ of $\mathcal{T}$ that goes through each edge e is bounded from above by the edge capacity $h(e)\lambda$. In this chapter, $f(e)$ is defined as $\sum_{T_i\in\mathcal{T}:e\in T_i}[\alpha+\beta q_{T_i}(e)]$, where $q_{T_i}(e)$ denotes the total demand that passes through the edge e along T_i. Term α means a fixed amount used to establish the routing T_i by separating the inside of T_i from the outside, while term $\beta q_{T_i}(e)$ means the net capacity proportional to the demand $q_{T_i}(e)$. The objective of GCTR is to construct the minimum cost network that admits a collection $\mathcal{T}$ of trees to send all demand to sink. This problem unifies CND, CMTR, and CTR. We prove that GCTR is $(2\lceil\lambda/(\alpha+\beta\kappa)\rceil/\lfloor\lambda/(\alpha+\beta\kappa)\rfloor+\rho_{\mathrm{ST}})$-approximable if $\lambda\geq\alpha+\beta\kappa$ holds. For GCTR instances with $\lambda<\alpha+\beta\kappa$, we construct a 13.037-approximation algorithm.

We also study a variant of GCTR in which it is allowed to purchase edge capacity in any required quantity. In this model, for each edge e of the underlying network, we assign capacity of $\lambda_e=\alpha|\mathcal{T}'|+\beta\sum_{T_i\in\mathcal{T}'}\sum_{v\in Z_i\cap D_{T_i}(v_i^e)}q(v)$ on e, where $\mathcal{T}'$ is the set of trees containing e. That is, the total cost of the constructed trees equals $\sum_{e\in E}\lambda_e w(e)$. We call this variant of GCTR, *the fractional generalized capacitated tree-routing problem* (FGCTR). We prove that *the fractional generalized capacitated tree-routing problem* (FGCTR) is 8.529-approximable.

Figure 1.4 summarizes problems studied in this thesis and suggests possible future work, where S denotes the set of sinks (or sources). For each edge in the figure, the problem given at its tail is a special case of that given at its head; the solid edges connect problems studied in the thesis, while the dashed edges refer to possible future work. Namely, it is left as a future work to define a multisink version of GCTR that unifies CCFL (to be defined in Chapter 3) and CMMTR. We may call such a problem *the generalized multisink capacitated tree-routing problem* (GMCTR for short). Note that MCEI is closely related to CND (to be defined in Chapter 3), where the difference is that CND allows the demand from a source to be split among different copies of the same edge. Therefore, it is also interesting to define a multisink version of MCEI which corresponds to CCFL (the multisink version of CND).

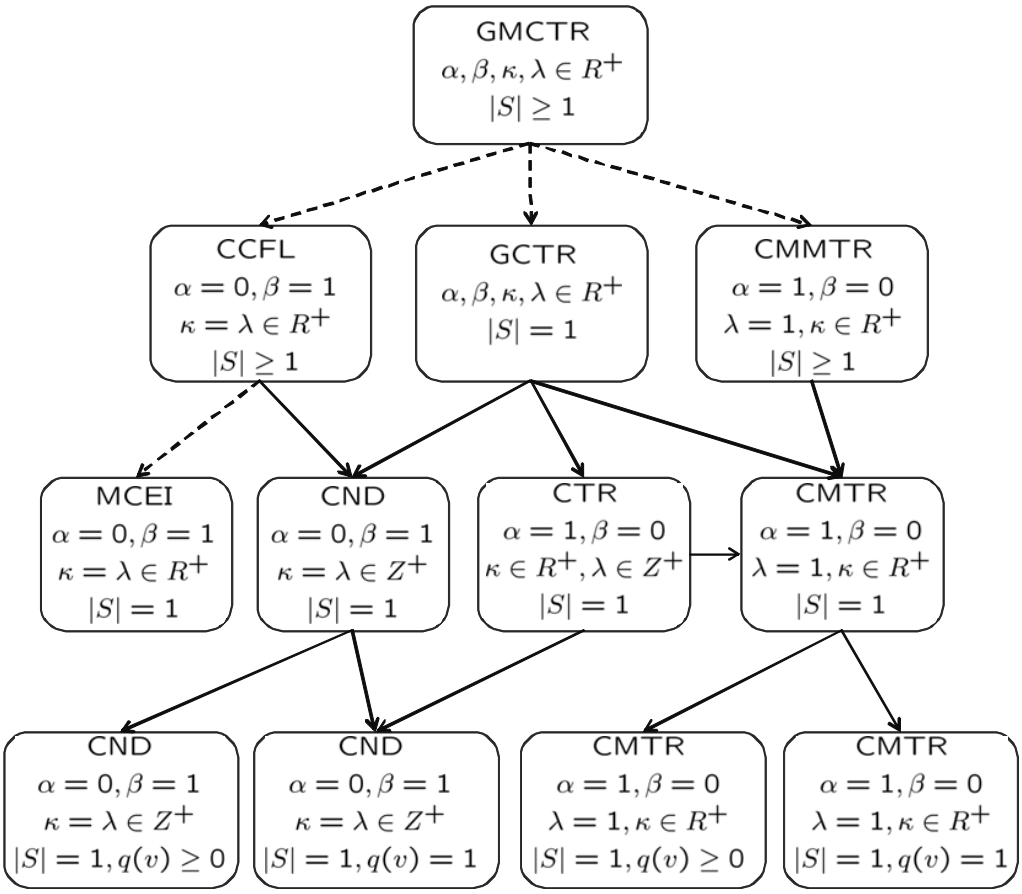

Figure 1.4: Illustration for the problems studied in this thesis and possible future work.

Chapter 2

The Capacitated Multicast Routing Problem in Networks

In this chapter, we present frameworks of approximation algorithms for the capacitated multicast tree routing problem and analyze its approximation ratio. Our algorithms are based on an elaborate tree cover of a tree.

2.1 Introduction

Multicast consists in sending a stream of data from a single source to multiple receivers or terminals, and is becoming increasingly popular in computer and communication networks supporting multimedia applications [36, 48, 70]. Multicast design is a more efficient method for supporting group communication than unicasting or broadcasting, because it allows transmission and routing of data packets to multiple destinations using fewer network resources.

In local area networks (LANs), terminals are connected through a broadcast network, and multicast in LANs is rather easily implemented. However, implementing multicast in wide area networks (WANs) is more complicated since the terminals are connected via switched/routed network [26]. In order to apply multicasting in WANs, the source node and all the terminals should be connected through a tree in the network [64]. Thus, the problem of finding a multicast routing in WANs is treated as a problem of constructing a multicast tree that spans the source and all terminals in the underlying network, where the goal is to minimize the cost of the multicast tree.

In this chapter, we study multicast under the multi-tree model, which has its origin in WDM optical networks with limited light-splitting capabilities [32]. Under this model, we are interested in constructing a set of trees of minimum total weight such that each tree spans the source node and a set of terminals of limited demand that are selected to receive data in the tree. In addition, every terminal in the underlying network is designated to receive data in exactly one of such trees. We call this problem the *capacitated multicast tree routing*

problem (CMTR for short), which is formally stated as follows.

Capacitated Multicast Tree Routing Problem (CMTR):
Input: A connected graph $G = (V, E)$, an edge weight function $w : E \to R^+$, a routing capacity $\kappa > 0$, a source $s \in V$, a set $M \subseteq V - \{s\}$ of terminals, and a demand function $q : M \to R^+$.
Feasible solution: A partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_\ell\}$ of M and a set $\mathcal{T} = \{T_1, T_2, \ldots, T_\ell\}$ of trees of G such that $Z_i \cup \{s\} \subseteq V(T_i)$ and $q(Z_i) \leq \kappa$ hold for each i. The number of copies of an edge $e \in E$ installed in the solution is given by $h_{\mathcal{T}}(e) = |\{T \in \mathcal{T} \mid e \in E(T)\}|$.
Goal: Minimize

$$\sum_{e \in E} h_{\mathcal{T}}(e) w(e) = \sum_{T_i \in \mathcal{T}} w(T_i).$$

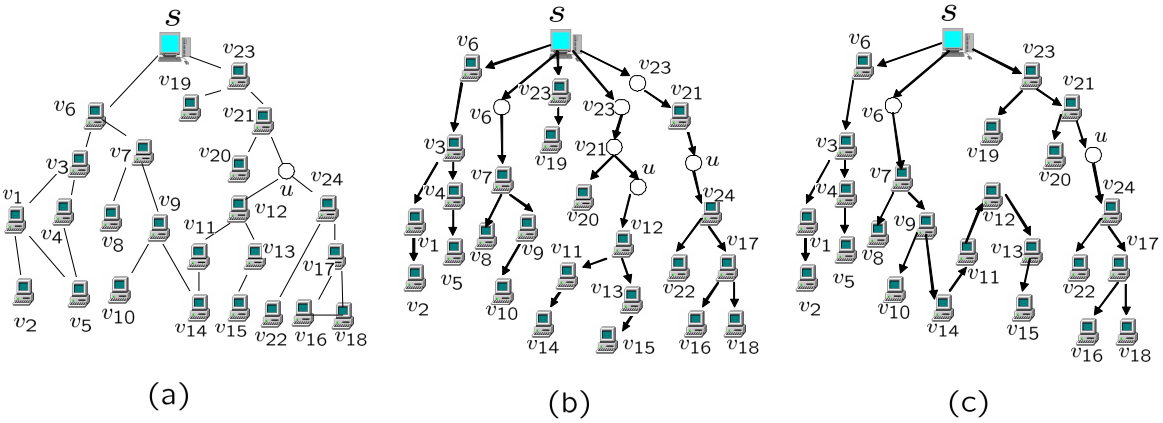

Figure 2.1: Illustration for CMTR; (a) an instance of CMTR; (b) a feasible solution to the instance in (a); (c) an optimal solution to the instance in (a).

Fig. 2.1(a) illustrates an instance of CMTR with $M = \{v_1, v_2, \ldots, v_{24}\}$, $q(v_i) = 1$ for all $v_i \in M$, $\kappa = 9$, and the weight of edge (v_9, v_{14}) equals 2 and all other edges have unit weights. Fig. 2.1(b) describes a feasible solution $(\mathcal{M}, \mathcal{T})$ to this instance, where $\mathcal{M} = \{Z_1 = \{v_1, v_2, \ldots, v_6\}, Z_2 = \{v_7, v_8, v_9, v_{10}\}, Z_3 = \{v_{19}, v_{23}\}, Z_4 = \{v_{11}, v_{12} \ldots, v_{15}, v_{20}\}, Z_5 = \{v_{16}, v_{17}, v_{18}, v_{21}, v_{22}, v_{24}\}\}$ and the set of branches of the tree in Fig. 2.1(b) forms $\mathcal{T}$. Fig. 2.1(c) describes an optimal solution $(\mathcal{M}^*, \mathcal{T}^*)$ to the instance in Fig. 2.1(a), where $\mathcal{M}^* = \{Z_1^* = \{v_1, v_2, \ldots, v_6\}, Z_2^* = \{v_7, v_8, \ldots, v_{15}\}, Z_3^* = \{v_{16}, v_{17}, \ldots, v_{24}\}\}$ and the set of branches of the tree in Fig. 2.1(c) forms $\mathcal{T}^*$.

The *unit demand case* of CMTR is the set of CMTR instances such that $q(v) = 1$ for all $v \in M$ and κ is a positive integer in an instance of CMTR. An instance of the unit demand case of CMTR may be written as (G, w, κ, s, M). Unit demand instances of CMTR with $\kappa = 1, 2$ can be solved optimally [25]. We assume that $\kappa \geq 3$ in unit demand instances of CMTR.

CMTR plays an important role in the design of telecommunication and optical networks as follows. To implement multicasting in a wavelength-routed optical network, the concept of a *light-tree* was proposed in [63]. Interconnecting the source and all terminals by a light-tree uses a dedicated wavelength on all of its branches. Each intermediate vertex in a light-tree must have a splitter so that copies of data can be made and delivered to each of its children. An n-way splitter is an optical device which splits an input signal into n outputs, thus reducing the power of each output to $(1/n)$th of that of the original signal. As a result, while the power budget may allow data on a given wavelength to be delivered to more than one terminal, it may not possible to deliver data to an arbitrary number of terminals using a single light-tree [32]. Hence, establishing a multicast connection in an optical network under the multi-tree model makes multicast easier and more efficient to implement at the expense of increasing the network cost. Under this model only a limited number of light splitting are allowed per transmission, and then a multicast routing is given as a set of light-trees such that each of them includes at most κ terminals, where parameter κ may be dependent on the size of routing vertices and the power budget of light transmission [26]. Therefore, each light-tree has at most $\lfloor \kappa/2 \rfloor$ intermediate vertices and each of them needs a κ-way splitter, implying that the signal from the source can be split at most $\lfloor \kappa/2 \rfloor$ times.

CMTR is closely related to the *capacitated minimum Steiner tree problem* (CMStT) studied recently in [40]. Given a connected graph $G = (V, E)$, an edge weight function $w : E \to R^+$, a positive real κ, a vertex $s \in V$, a set $M \subseteq V - \{s\}$ of terminals, and a vertex weight function $q : M \to R^+$, CMStT consists in finding a minimum Steiner tree T spanning s and all terminals such that the total vertex weight in the descendant of each child of s in T is at most κ. In particular, any feasible solution to CMStT is a feasible solution to CMTR. When $M = V$, this problem is known as the *capacitated minimum spanning tree problem* (CMST).

CMTR is proven to be NP-hard [26]. CMTR has received a number of attentions in the recent study. A $(2 + \rho_{\mathrm{ST}})$-approximation algorithm to CMTR with a general demand can be obtained by modifying the algorithm due to Jothi and Raghavachari [40] designed for CMStT, where ρ_{ST} is any achievable approximation ratio for finding a minimum cost Steiner tree on $M \cup \{s\}$. In the next section, we present details of the algorithm and its analysis. Note that, Lin [49] showed that the unit demand case of CMTR remains NP-hard even if $\kappa = 3$. For the unit demand case of CMTR, there have been developed several constant-factor approximation algorithms. Based on Hamilton circuit, Gu et al. [26] proposed a 4-approximation algorithm. Lin [49] gave a $(2.4 + \rho_{\mathrm{ST}})$-approximation algorithm. Afterwards Cai et al. [10] gave a $(2 + \rho_{\mathrm{ST}})$-approximation algorithm. If the vertex set V consists of points in the L_p metric plane, then a $(3/2 + (7/5)\rho_{\mathrm{ST}})$-approximation algorithm to CMTR is proposed by Jothi and Raghavachari [40] which is designed for CMStT instances with unit vertex weights. Recently, Cai et al. [11] proved a $(8/5 + (5/4)\rho_{\mathrm{ST}})$-approximation algorithm which improves the previous best approximation ratio of $(2 + \rho_{\mathrm{ST}})$ for the best known ratio

of $\rho_{\rm ST} = 1 + \frac{\ln 3}{2} < 1.55$.

In this chapter, we propose a $(2 + \rho_{\rm ST})$-approximation algorithm and a $(3/2 + (4/3)\rho_{\rm ST})$-approximation algorithm to the general and unit demand cases of CMTR, respectively. Our algorithm for the unit demand case outperforms the $(3/2+(7/5)\rho_{\rm ST})$-approximation algorithm which is designed for the L_p metric in the plane. Note that the $(8/5+(5/4)\rho_{\rm ST})$-approximation algorithm due to Cai et al. [11] improves over $(3/2 + (4/3)\rho_{\rm ST})$ as long as $\rho_{\rm ST} \geq 1.2$. In particular, it is known that $\rho_{\rm ST} = 1$ when $M = V$ since the Steiner tree problem with terminal set $M = V$ in G becomes the minimum spanning tree problem. Hence our approximation ratio improves that obtained by Cai et al. [11] in the case where $M = V$.

Note that, if the vertex set V consists of points in the L_p metric in the plane, then our algorithm for the unit demand case of CMTR can be slightly modified to provide a $(3/2 + (4/3)\rho_{\rm ST})$-approximation algorithm to the unit vertex weight case of CMStT which gives the best known approximation ratio for the unit demand case of CMST (i.e., in the case of $\rho_{\rm ST} = 1$).

Given an instance $I = (G, w, \kappa, s, M, q)$ of CMTR, our algorithm first produces a tree T of minimum cost including all vertices in $M \cup \{s\}$, then breaks T into a set of subtrees each of which contains a set of terminals with at most κ demands, and finally connects each of such subtrees to s.

Note that the high-level description of our algorithm for the unit demand case is analogous to that in [10], [11], [40], and [49] but with different tree cover techniques. From the analysis in [10], [40], and [49], we can see that one ingredient for improving the approximation ratio for CMTR is to design a tree cover for T such that (i) the number of terminals specified for each of the obtained trees is as close as possible to κ, and (ii) the total cost of these trees is minimized. In this chapter, we design a tree cover for T that achieves almost the same average on the cardinality (the number of terminals) of each tree as that in [40], but with less cost. On the other hand, the average on the cardinality of each tree in our tree cover is greater than that in [10] and [49] at the expense of increasing the total cost to at most (4/3) of that in [10] and [49]. The algorithm due to Cai et al. [11] constructs a set of trees with less total tree weights and each of which has more terminals than ours. The running times of our algorithm and algorithms in [10], [11], [40], and [49] are dominated by the approximation algorithm for the Steiner tree problem.

The following lower bound on the optimal value of CMTR has been proved and used to derive approximation algorithms to the unit demand case of CMTR [10, 11, 25, 26, 40, 49].

Lemma 2.1. *For an instance $I = (G, w, \kappa, s, M, q)$ of CMTR, let $opt(I)$ be the weight of an optimal solution $(\mathcal{M}^*, \mathcal{T}^*)$ to I, T^* be the minimum weight of a tree that spans $M \cup \{s\}$ in G, and $d(t)$, $t \in M$, be the distance from s to t. Then*

$$\max\left\{ w(T^*), \frac{1}{\kappa} \sum_{v \in M} q(v)d(v) \right\} \leq opt(I).$$

Proof. We first prove that $w(T^*) \leq opt(I)$.

The set of all edges used in the optimal solution ($\mathcal{M}^* = \{Z_1, \ldots, Z_\ell\}, \mathcal{T}^* = \{T_1, \ldots, T_\ell\}$) is given by

$$E(\mathcal{T}^*) = \cup_{T_i \in \mathcal{T}^*} E(T_i).$$

Clearly, the edge set $E(\mathcal{T}^*)$ contains a tree T that spans $M \cup \{s\}$ in G, and the weight $w(T)$ of T is at most that of CMTR solution. Then the weight $w(T^*)$ of T^* is at most the optimal value to CMTR instance I.

We next prove that

$$\sum_{v \in M} q(v)d(v) \leq \kappa \cdot opt(I).$$

Note that $q(Z_i) \leq \kappa$ holds for all $Z_i \in \mathcal{M}^*$, and hence

$$\kappa \cdot opt(I) \quad = \quad \kappa \sum_{T_i \in \mathcal{T}^*} w(T_i) \geq \sum_{T_i \in \mathcal{T}^*} q(Z_i)w(T_i) \geq \sum_{v \in M} q(v)d(v),$$

since $d(v) \leq w(T_i)$ for all vertices v in T_i. □

2.2 General Demands

This section introduces a "balanced" partition of a set of terminals, which provides an approximation solution to CMTR.

For a tree T rooted at a vertex r, an ordered partition $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_p\}$ of a subset of the terminal set M is called *κ-balanced* if the following holds:

(i) $q(Z_i) \leq \kappa$ for $i = 1, 2, \ldots, p$;

(ii) $q(Z_i) > \kappa/2$ for $i = 1, 2, \ldots, p-1$, and if $p \geq 2$ then $q(Z_{p-1} \cup Z_p) > \kappa$; and

(iii) Each $T\langle Z_j \rangle$ $(j = 1, 2, \ldots, p-1)$ has no common edge with $T\langle \cup_{j<i\leq p} Z_i \cup \{r\} \rangle$.

Lemma 2.2. *There always exists a κ-balanced partition if* $\max_{v \in M} q(v) \leq \kappa$.

Proof. First of all, we assume for simplicity and without loss of generality that in a given tree T, (i) all terminals are leaves, i.e., $M = L(T)$, by introducing a new edge of weight zero for each non-leaf terminal, and (ii) $|Ch_T(v)| = 2$ holds for every non-leaf $v \in V(T)$, i.e., T is a binary tree rooted at r, by replicating internal vertices of degree more than 3, so that the copies of the same vertex are connected with zero-weight edges. It suffices to show that the lemma holds for such a binary tree T.

Then a κ-balanced partition of M can be obtained by repeating the following procedure as long as the total demand of the current tree is more than κ: choose a vertex v with the maximum depth in the current tree such that $q(V(T_v) \cap M) > \kappa/2$ and delete T_v from the current tree after letting the terminal set of T_v be the next new subset Z_i. Note that $q(Z_i) \leq \kappa$ since $q(V(T_u) \cap M) \leq \kappa/2$ holds for each child $u \in Ch(v)$ by the choice of v. Moreover, it

is easy to observe that $T\langle Z_i\rangle$ has no common edges with the current tree. Finally, let Z_p be the terminal set of the remaining tree after the last iteration. Then it hold $q(Z_p) \leq \kappa$ and $q(Z_{p-1} \cup Z_p) > \kappa$ by the choice of v if there was at least one iteration of the procedure, i.e., $p \geq 2$. This proves the lemma. □

Based on κ-balanced partition, we obtain an approximation algorithm for CMTR with general demand. The basic idea of the algorithm is to compute an approximate Steiner tree T in $(G, w, M \cup \{s\})$, regard T as a tree rooted at s, and then find a κ-balanced partition $\mathcal{Z}$ of M in T. For each $Z \in \mathcal{Z}$, we choose a vertex $t_Z \in Z$ and connect the tree $T\langle Z\rangle$ to s by adding a shortest path between s and t_Z in (G, w), where we call such a vertex t_Z the *hub vertex* of Z. We describe the algorithm in the following form which will be also used in Chapter 5.

Algorithm GeneralCMTR
Input: A CMTR instance $I = (G, w, \kappa, s, M, q)$.
Output: A solution $(\mathcal{M}, \mathcal{T})$ to I.

Step 1. Compute a ρ_{ST}-approximate solution T to the Steiner tree problem in (G, w) that spans $M \cup \{s\}$ and then regard T as a tree rooted at s.
Define a vertex weight function $d : M \to R^+$ by setting

$$d(v) := d_{(G,w)}(s, v), \quad v \in M.$$

Step 2. Find a partition $\mathcal{M}$ of M.
For each subset $Z \in \mathcal{M}$, assign a vertex $t_Z \in V(T)$ as its hub vertex.
Let S be the set of all hub vertices.

Step 3. For each hub vertex $t \in S$, we choose a shortest path $SP(s, t)$ between s and t in (G, w). For each subset $Z \in \mathcal{M}$, let T_Z be the tree obtained from $T\langle Z \cup \{t_Z\}\rangle$ by adding the edge set in $SP(s, t_Z)$. Let $\mathcal{T} := \{T_Z \mid Z \in \mathcal{M}\}$. □

For a CMTR instance with general demand, we realize Step 2 as follows. We compute a κ-balanced partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_p\}$ of M. For $j = 1, 2, \ldots, p-1$, we choose a terminal $t_{Z_j} \in Z_j$ with the minimum distance $d(t_{Z_j})$ as its hub vertex, and let $t_{Z_p} := s$.

Theorem 2.1. *Given a* CMTR *instance* $I = (G, w, \kappa, s, M, q)$, *algorithm* GeneralCMTR *with the above Step 2 delivers a* $(2 + \rho_{\mathrm{ST}})$*-approximate solution to* I.

Proof. By Property (iii) of κ-balanced partition, each edge in T is used at most once in the union of subtrees in $\mathcal{T}' = \{T\langle Z_j\rangle \mid j = 1, 2, \ldots, p-1\} \cup \{T\langle Z_p \cup \{s\}\rangle\}$. Note that $\mathcal{T}' = \{T\langle Z \cup \{t_Z\}\rangle \mid Z \in \mathcal{M}\}$ holds by the choice of hub vertices. Therefore, the total weight of the edges to be installed for constructing $\mathcal{T}$ is bounded by the weight of T plus the sum of the shortest paths used; i.e., it holds

$$\sum_{e \in E} h_{\mathcal{T}}(e) w(e) \leq w(T) + \sum_{t \in S} d(t). \tag{2.1}$$

For a minimum Steiner tree T^* that spans $M \cup \{s\}$, we have $w(T^*) \leq opt(I)$ by Lemma 2.1. Hence $w(T) \leq \rho_{\mathrm{ST}} \cdot w(T^*) \leq \rho_{\mathrm{ST}} \cdot opt(I)$ holds. To prove the theorem, it suffices to show that

$$\sum_{t \in S} d(t) \leq 2opt(I). \tag{2.2}$$

The choice of hub vertices and Property (ii) of κ-balanced partition imply that, for each $Z_i \in \mathcal{M}$, $i = 1, 2, \ldots, p-1$, we have

$$\sum_{v \in Z_i} q(v)d(v) \geq d(t_{Z_i}) \sum_{v \in Z_i} q(v) > d(t_{Z_i})\kappa/2. \tag{2.3}$$

By summing inequality (2.3) overall $Z_i \in \mathcal{M}$, $i = 1, 2, \ldots, p-1$, we have

$$(1/2)\sum_{t \in S} d(t) < \sum_{1 \leq i \leq p-1} \sum_{v \in Z_i} q(v)d(v)/\kappa \leq \sum_{v \in M} q(v)d(v)/\kappa.$$

By Lemma 2.1, this proves (2.6). □

In the rest of this chapter we study the unit demand case of CMTR. We start by some results on tree covers in a tree, based on which our approximation algorithm is built.

2.3 Tree Cover

This section describes how to construct a "tree cover" in an edge-weighted tree. A tree cover is a collection of subtrees that covers all vertices, in which two objectives must be taken into consideration, (i) the number of terminals in each of the obtained trees is as close as possible to a specified integer κ, and (ii) the total cost of these trees is minimized. Such a tree cover will be the basis of our approximation algorithm given in Section 2.4. We first present some results for special cases of tree covers.

2.3.1 Tree covers in special cases

In this subsection, we prepare several lemmas on tree covers for a tree with a special structure. We first introduce a subgraph which plays a key role in our algorithm.

Definition 2.1. *For a vertex v in a rooted tree T, a terminal set $Z_v \subseteq V(T_v) - \{v\}$, and a positive integer κ, a binary rooted tree T_v is said to be a* 2/3-balance-tree *if $|Z_v| > \kappa$ holds and the total number of terminals in each of the branches of T_v is less than $(2/3)\kappa$.*

For a tree T_x with a terminal set Z_x, the following lemma partitions Z_x into two subsets such that either, (i) the cardinality of each subset lies between $(2/3)\kappa$ and κ, or (ii) each of which has at most κ terminals and a nonempty intersection with a subset $Z_0 \subseteq Z_x$ with $|Z_0| \geq (2/3)\kappa$. Such a partition can be obtained at the expense of increasing the total cost by at most $(1/3)w(T_x)$.

Lemma 2.3. *Let κ be a positive integer and T_x be a rooted tree with a terminal set $Z_x \subseteq V(T_x) - \{x\}$ such that $(4/3)\kappa \leq |Z_x| \leq 2\kappa$. Suppose that T_x consists of three branches B_1, B_2 and B_3 such that $T_x - B_1$ is a 2/3-balance-tree rooted at x. Then there is a partition $\pi = \{X, Y\}$ of Z_x such that $w(T\langle X\rangle) + w(T\langle Y\rangle) \leq (4/3)w(T_x)$ and one of the following holds:*

(i) $(2/3)\kappa \leq |X|, |Y| \leq \kappa$.

(ii) *For any specified subset $Z_0 \subseteq Z_x$ with $|Z_0| \geq (2/3)\kappa$, it hold* $\max\{|X|, |Y|\} \leq \kappa$ *and* $X \cap Z_0 \neq \emptyset \neq Y \cap Z_0$.

Proof. Let $Z_i = V(B_i) \cap Z_x$, $i = 1, 2, 3$. Note that $(1/3)\kappa < |Z_2|, |Z_3| < (2/3)\kappa$ and $\kappa < |Z_2| + |Z_3| < (4/3)\kappa$ by the definition of a 2/3-balance-tree. We have $|Z_1| = |Z_x| - (|Z_2| + |Z_3|) < 2\kappa - \kappa = \kappa$. The main idea of the proof is to partition the elements of the lightest branch into two appropriate sets and to combine them with the remaining two branches. Note that the smallest weight of $w(B_1)$, $w(B_2)$, and $w(B_3)$ is at most $(1/3)w(T_x)$. Assume without loss of generality that $w(B_2)$ attains the smallest weight. We distinguish the following two cases.

Case 1. $|Z_1| \geq (2/3)\kappa$: To show that (i) holds in this case, we partition the elements of B_2 into two sets of appropriate cardinalities and combine them with B_1 and B_3. We have $(5/3)\kappa = (2/3)\kappa + \kappa < |Z_x| = |Z_1| + (|Z_2| + |Z_3|) \leq 2\kappa$. Choose a subset $F \subseteq Z_2$ with cardinality $|F| = \lceil |Z_x|/2 \rceil - |Z_1|$. Note that $\lceil |Z_x|/2 \rceil - |Z_1| \leq (2\kappa/2) - (2/3)\kappa = (1/3)\kappa < |Z_2|$ and $\lceil |Z_x|/2 \rceil - |Z_1| \geq (|Z_2| + |Z_3| - |Z_1|)/2 > (\kappa - \kappa)/2 = 0$ hold, since $(2/3)\kappa \leq |Z_1| < \kappa$, $|Z_x| \leq 2\kappa$, and $|Z_2| + |Z_3| > \kappa$. Thus F is well defined. Let $X := Z_1 \cup F$ and $Y := Z_3 \cup (Z_2 - F) = Z_x - X$. Hence $|X| = |Z_1| + |F| = \lceil |Z_x|/2 \rceil$ and $|Y| = |Z_x| - |X| = \lfloor |Z_x|/2 \rfloor$. We have $(2/3)\kappa \leq |X| \leq \kappa$ and $(2/3)\kappa < |Z_x|/2 - 1/2 \leq |Y| \leq \kappa$ (since $(5/3)\kappa < |Z_x| \leq 2\kappa$ and $\kappa \geq 3$).

Case 2. $|Z_1| < (2/3)\kappa$: We show that (ii) holds in this case. Choose an arbitrary subset $Z_0 \subseteq Z_x$ with cardinality $|Z_0| \geq (2/3)\kappa$. Note that $|Z_1|, |Z_2|, |Z_3| < (2/3)\kappa$ and $|Z_0| \geq (2/3)\kappa$ imply that Z_0 contains terminals from at least two subsets of Z_1, Z_2, and Z_3. We partition the terminals of B_2 into two appropriate subsets and combine them to B_1 and B_3 such that each subset of the resulting partition contains elements from Z_0. In particular, we choose a subset F of Z_2 such that $X = Z_1 \cup F$ and $Y = Z_x - X$ satisfy $\max\{|X|, |Y|\} \leq \kappa$ and $X \cap Z_0 \neq \emptyset \neq Y \cap Z_0$. □

Consider the tree T_x shown in Fig. 2.2(a), with a terminal set Z_x, where $|V(B_1) \cap Z_x|, |V(B_2) \cap Z_x| < (2/3)\kappa$ and $|V(B') \cap Z_x|, |V(B_4) \cap Z_x| \leq (1/3)\kappa$. Given a subset $Z_0 \subseteq Z_x$ with $|Z_0| \geq (2/3)\kappa$, the next lemma partitions T_x into two subtrees each of which has at most κ terminals and a nonempty intersection with Z_0.

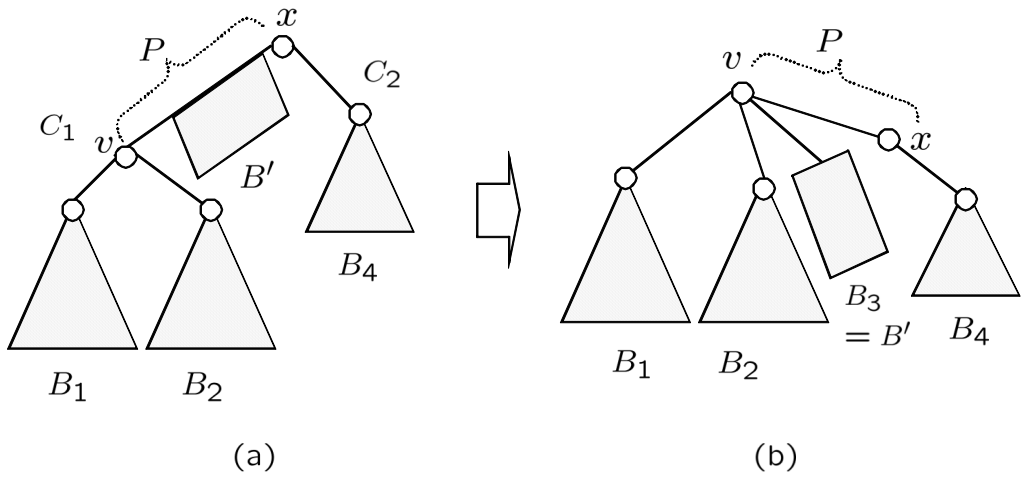

Figure 2.2: Illustration for Lemma 2.4; (a) a tree T_x defined in Lemma 2.4; (b) a tree T'_v obtained from T_x by duplicating P.

Lemma 2.4. *Let κ be a positive integer and T_x be a binary rooted tree with a terminal set $Z_x \subseteq V(T_x) - \{x\}$. Let C_1 and C_2 be the two branches of T_x such that C_1 contains a 2/3-balance-tree T_v with $v \neq x$ and satisfies $\kappa < |Z_x \cap V(C_1)| < (4/3)\kappa$, and $|Z_x \cap V(C_2)| \leq (1/3)\kappa$. Then for any subset $Z_0 \subseteq Z_x$ with $|Z_0| \geq (2/3)\kappa$, there is a partition $\{B, C\}$ of Z_x such that $\max\{|B|, |C|\} \leq \kappa$, $B \cap Z_0 \neq \emptyset \neq C \cap Z_0$, and $w(T\langle B\rangle) + w(T\langle C\rangle) \leq w(T_x) + w(B')$ for the tree B' obtained from C_1 deleting vertices in $D(v) - \{v\}$* (see Fig. 2.2(a)).

Proof. There is a unique edge (x, y) with $y \in V(C_1)$ in T_x. Let P denote the path from x to v in T_x. Note that $w(P) \leq w(B')$. To find a desired partition $\{B, C\}$ of Z_x, we transform T_x into another tree T'_v by removing edge (x, y) and adding a new edge (x, v) of weight $w(P)$. We regard T'_v as a tree rooted at v (Fig. 2.2(b) illustrates how a tree T'_v is constructed from T_x described in Fig. 2.2(a)). Note that $|Z_x \cap V(B')| < (1/3)\kappa$ since $|Z_x \cap V(C_1)| < (4/3)\kappa$ and $|Z_x \cap V(T_v)| > \kappa$. Hence T'_v has exactly two branches each of which has less than $(2/3)\kappa$ and more than $(1/3)\kappa$ terminals and two branches each of which has at most $(1/3)\kappa$ terminals. Moreover, $w(T'_v) \leq w(T_x) + w(B')$ holds.

Let B_1 and B_2 be the large branches of T'_v, and B_3 and B_4 be the small branches of T'_v. Let $Z_i = V(B_i) \cap Z_x$, $i = 1, 2, 3, 4$. Since $|Z_i| < (2/3)\kappa$ for all i and $|Z_0| \geq (2/3)\kappa$, at least two branches of T'_v contain terminals from Z_0. Let $B = Z_i \cup Z_j$ and $C = Z_{i'} \cup Z_{j'}$, where $\{i, i'\} = \{1, 2\}$ and $\{j, j'\} = \{3, 4\}$ such that $B \cap Z_0 \neq \emptyset$ and $C \cap Z_0 \neq \emptyset$ hold. Hence $\max\{|B|, |C|\} \leq \kappa$ holds since $\max\{|Z_1|, |Z_2|\} < (2/3)\kappa$ and $\max\{|Z_3|, |Z_4|\} \leq (1/3)\kappa$. By construction of T'_v from T_x, we have $w(T\langle B\rangle) + w(T\langle C\rangle) \leq w(T'_v) \leq w(T_x) + w(B')$. □

The following lemma partitions a tree T_x into three subtrees such that either, (i) the cardinality of each subtree lies between $(2/3)\kappa$ and κ (see Fig. 2.3(b)), or (ii) the cardinality of one subtree lies between $(2/3)\kappa$ and κ and the other two are obtained by applying Lemma 2.4 to the subtree on the remaining terminals (see Fig. 2.3(c)).

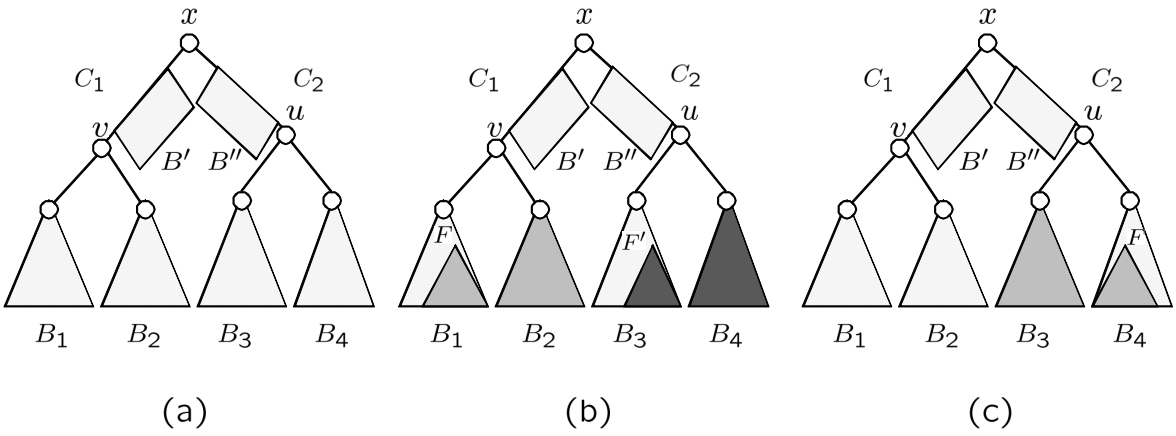

Figure 2.3: Illustration for Lemma 2.5; (a) the construction of the tree T_x defined in Lemma 2.5; (b) illustration for Case 1 in Lemma 2.5; (c) illustration for Case 2 in Lemma 2.5.

Lemma 2.5. *Let κ be a positive integer and T_x be a binary rooted tree with a terminal set $Z_x \subseteq V(T_x) - \{x\}$. Let C_1 and C_2 be the two branches of T_x such that $\kappa < |Z_x \cap V(C_i)| < (4/3)\kappa$, $i = 1, 2$. Suppose that C_1 and C_2 contain 2/3-balance-trees T_v and T_u, respectively. Then there is one of the following partitions π of Z_x:*

(i) *$\pi = \{A, B, C\}$ such that $(2/3)\kappa \leq |A|, |B|, |C| \leq \kappa$ and $w(T\langle A\rangle)+w(T\langle B\rangle)+w(T\langle C\rangle) \leq (4/3)w(T_x)$.*

(ii) *$\pi = \{A, \overline{A}\}$ such that $(2/3)\kappa \leq |A| \leq \kappa$ and $(4/3)\kappa \leq |\overline{A}| < (5/3)\kappa$. Moreover, for any specified subset $Z_0 \subseteq \overline{A}$ with $|Z_0| \geq (2/3)\kappa$, $\overline{A}$ can be partitioned into B and C such that $\max\{|B|, |C|\} \leq \kappa$, $B \cap Z_0 \neq \emptyset \neq C \cap Z_0$, and $w(T\langle A\rangle) + w(T\langle B\rangle) + w(T\langle C\rangle) \leq (4/3)w(T_x)$.*

Proof. Let $Z_{C_i} = Z_x \cap V(C_i)$, $i = 1, 2$. Let B' (resp., B'') denote the tree obtained from C_1 (resp., C_2) deleting vertices in $D(v) - \{v\}$ (resp., $D(u) - \{u\}$). Let $Z_v = Z_x \cap V(T_v)$, $Z_u = Z_x \cap V(T_u)$, $Z' = Z_x \cap V(B')$, and $Z'' = Z_x \cap V(B'')$. Note that $|Z'| < (1/3)\kappa$ since $|Z_{C_1}| < (4/3)\kappa$ and $|Z_v| > \kappa$. Similarly $|Z''| < (1/3)\kappa$. Denote the two branches of T_v (resp., T_u) by B_1 and B_2 (resp., B_3 and B_4). See Fig. 2.3(a) for the construction of T_x. Let $Z_i = Z_x \cap V(B_i)$, $i = 1, 2, 3, 4$, where $|Z_i| > (1/3)\kappa$ holds. Note that the smallest weight of $w(B_1) + w(B_3), w(B_4) + w(B')$, and $w(B_2) + w(B'')$ is at most $(1/3)w(T_x)$. We distinguish two different cases.

Case 1. $w(B_1) + w(B_3)$ attains the smallest weight: To show that (i) holds in this case, we partition the elements of B_1 (resp., B_3) into two sets of appropriate cardinalities and combine them with $B'+B''$ and B_2 (resp., B_4) (see Fig. 2.3(b)). Namely, we choose two subsets $F \subseteq Z_1$ and $F' \subseteq Z_3$ of cardinalities $|F| = \lceil (1/3)\kappa \rceil - |Z'|$ and $|F'| = \lceil (1/3)\kappa \rceil - |Z''|$, respectively. Note that F and F' are well defined since $\min\{|Z_1|, |Z_3|\} > (1/3)\kappa$ and $\max\{|Z'|, |Z''|\} <$

$(1/3)\kappa$ imply that $0 < \lceil(1/3)\kappa\rceil - |Z'| \le |Z_1|$ and $0 < \lceil(1/3)\kappa\rceil - |Z''| \le |Z_3|$. The desired partition can be obtained by setting $A := Z_{C_1} - (F \cup Z')$, $B := Z_{C_2} - (F' \cup Z'')$, and C to be the remaining elements of Z_x. By the choice of F and F', we have $(2/3)\kappa < |A|, |B| < \kappa$ and $|C| = |F| + |F'| + |Z'| + |Z''| \ge (2/3)\kappa$. If $\kappa = 3$, then $|C| = 2 < \kappa$. Otherwise ($\kappa \ge 4$), $|C| = 2\lceil(1/3)\kappa\rceil \le 2((1/3)\kappa + 2/3) \le \kappa$. Moreover, $w(B_1) + w(B_3) \le (1/3)w(T_x)$ gives the desired upper bound on $w(T\langle A\rangle) + w(T\langle B\rangle) + w(T\langle C\rangle)$.

Case 2. $w(B_4) + w(B')$ or $w(B_2) + w(B'')$ attains the smallest weight; $w(B_4) + w(B') \le w(B_2) + w(B'')$ is assumed without loss of generality: The main idea of the proof of this case is to partition the elements of B_4 into two sets of appropriate cardinalities and combine them with $T_x - T_u$ and B_3 (see Fig. 2.3(c)). Choose a subset $F \subseteq Z_4$ with cardinality $|F| = \lfloor(1/3)\kappa\rfloor - |Z''|$. Note that F is well defined since $|Z_4| > (1/3)\kappa$ and $|Z''| < (1/3)\kappa$ imply that $0 \le \lfloor(1/3)\kappa\rfloor - |Z''| < |Z_4|$. Let $A = Z_{C_2} - (F \cup Z'')$ and $\overline{A} = Z_x - A$. The choice of F implies that $(2/3)\kappa < |A| = |Z_{C_2}| - \lfloor(1/3)\kappa\rfloor \le \kappa$ and $(4/3)\kappa < |\overline{A}| = |Z_{C_1}| + \lfloor(1/3)\kappa\rfloor < (5/3)\kappa$. Moreover, it holds

$$w(T\langle A\rangle) + w(T\langle\overline{A}\rangle) \le w(T_x) + w(B_4). \tag{2.4}$$

By regarding $T\langle\overline{A}\rangle$ as a tree rooted at x, we see that $T\langle\overline{A}\rangle$ satisfies the conditions in Lemma 2.4. Choose an arbitrary subset $Z_0 \subseteq \overline{A}$ with $|Z_0| \ge (2/3)\kappa$. Apply Lemma 2.4 to $T\langle\overline{A}\rangle$ to get a partition $\{B, C\}$ of $\overline{A}$ such that $\max\{|B|, |C|\} \le \kappa$, $B \cap Z_0 \ne \emptyset \ne C \cap Z_0$, and

$$w(T\langle B\rangle) + w(T\langle C\rangle) \le w(T\langle\overline{A}\rangle) + w(B'). \tag{2.5}$$

From (2.4) and (2.5), we get $w(T\langle A\rangle) + w(T\langle B\rangle) + w(T\langle C\rangle) \le w(B_4) + w(B') + w(T_x) \le (4/3)w(T_x)$. □

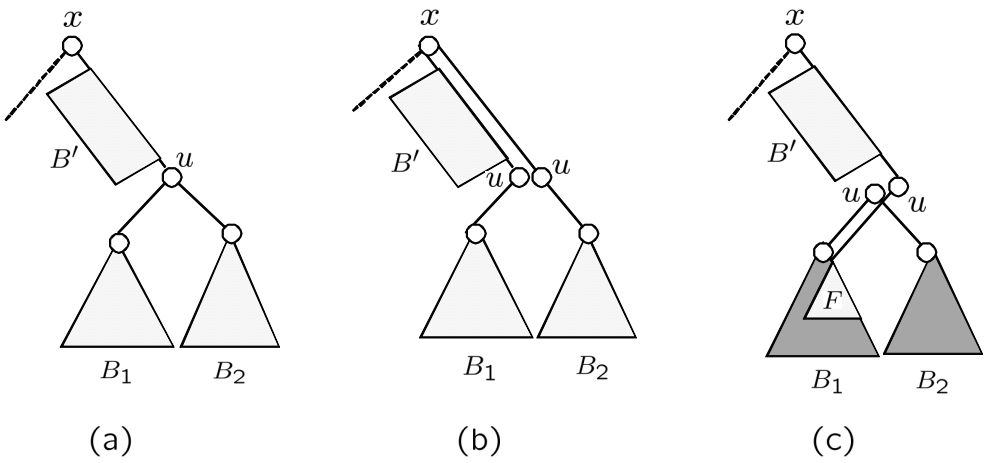

Figure 2.4: Illustration of Lemma 2.6; (a) a branch B of T_x such that T_u is a 2/3-balance-tree and $\kappa < |Z_x \cap V(B)| < (4/3)\kappa$; (b) $w(B') \le \min\{w(B_1), w(B_2)\}$; (c) $w(B_1) \le \min\{w(B_2), w(B')\}$.

In the following lemma, we partition a special tree T_x into a set of subtrees such that, (i) each subtree has at most κ terminals, and (ii) each subtree not containing x has at least $(2/3)\kappa$ terminals.

Lemma 2.6. *Let κ be a positive integer and T_x be a binary rooted tree with a terminal set $Z_x \subseteq V(T_x) - \{x\}$ such that for each $u \in Ch(x)$, if $|D(u) \cap Z_x| \geq (2/3)\kappa$, then $T\langle D(u) \cap Z_x\rangle$ contains a 2/3-balance-tree and satisfies $\kappa < |D(u) \cap Z_x| < (4/3)\kappa$. Then there is a partition $\mathcal{Z}_1 \cup \mathcal{Z}_2$ of Z_x such that*

(i) *$(2/3)\kappa \leq |Z| \leq \kappa$ for each subset $Z \in \mathcal{Z}_1$;*

(ii) *$|Z| < \kappa$ for each subset $Z \in \mathcal{Z}_2$; and*

(iii)
$$\sum_{Z \in \mathcal{Z}_1} w(T\langle Z\rangle) + \sum_{Z \in \mathcal{Z}_2} w(T\langle Z \cup \{x\}\rangle) \leq (4/3)w(T_x).$$

Proof. For each branch B of T_x with less than $(2/3)\kappa$ terminals, we add the set of terminals of B to $\mathcal{Z}_2$. Now we consider a branch B of T_x with $\kappa < |Z_x \cap V(B)| < (4/3)\kappa$, which contain a 2/3-balance-tree. Denote the 2/3-balance-tree of B by T_u and its branches by B_1 and B_2. Let B' denote the tree obtained from B deleting vertices in $D(u) - \{u\}$ (Fig. 2.4(a) illustrates the construction of B). Let $Z = Z_x \cap V(B)$, $Z_i = Z_x \cap V(B_i)$, $i = 1, 2$, $Z_u = Z_x \cap V(T_u)$, and $Z' = Z_x \cap V(B')$. Note that $|Z'| = |Z| - |Z_u| < (4/3)\kappa - \kappa = (1/3)\kappa$.

Now we partition the elements of B into two subsets with appropriate cardinalities depending on the smallest weight among $w(B_1), w(B_2)$, and $w(B')$, and add them to one of $\mathcal{Z}_1$ and $\mathcal{Z}_2$. Note that the smallest weight of $w(B_1), w(B_2)$, and $w(B')$ is at most $(1/3)w(B)$. If $w(B') \leq \min\{w(B_1), w(B_2)\}$, then let $X = Z_1 \cup Z'$ and $Y = Z_2$. Add X and Y to $\mathcal{Z}_2$. Note that $|X| = |Z_1| + |Z'| < (2/3)\kappa + (1/3)\kappa = \kappa$. Moreover, $w(T\langle X \cup \{x\}\rangle) + w(T\langle Y \cup \{x\}\rangle) \leq w(B) + w(B') \leq (4/3)w(B)$ holds (see Fig. 2.4(b)). Otherwise, assume without loss of generality that $w(B_1) \leq w(B_2)$. Choose a subset $F \subseteq Z_1$ of cardinality $|F| = \lceil (1/3)\kappa \rceil$, and let $X = Z_u - F$ and $Y = Z - X$. Such a subset F is well defined since $|Z_1| > (1/3)\kappa$. Note that $(2/3)\kappa < \kappa - \lceil (1/3)\kappa \rceil < |X| < (4/3)\kappa - \lceil (1/3)\kappa \rceil \leq \kappa$ and $|Y| = |Z| - |X| < (2/3)\kappa$. Add X and Y to $\mathcal{Z}_1$ and $\mathcal{Z}_2$, respectively. $w(B_1) \leq (1/3)w(B)$ implies that $w(T\langle X\rangle) + w(T\langle Y \cup \{v\}\rangle) \leq (4/3)w(B)$ (see Fig. 2.4(c)). □

2.3.2 Algorithm for tree cover

In this subsection, we show that, for an arbitrary tree, there exists a tree cover given in the next theorem. For this, we present an algorithm that exploits the results in the previous subsection to compute such a tree cover.

Lemma 2.7. *Given a tree T rooted at a vertex s, an edge weight function $w : E(T) \to R^+$, a positive integer κ, a terminal set $M \subseteq V(T)$, and a vertex weight function $d : M \to R^+$, there is a partition $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$ of M, where $\mathcal{M}_3 = \{X_1, Y_1, \ldots, X_r, Y_r\}$, that satisfies:*

(i) *$|Z| < \kappa$ for each terminal subset $Z \in \mathcal{M}_1$.*

(ii) *$(2/3)\kappa \leq |Z| \leq \kappa$ for each terminal subset $Z \in \mathcal{M}_2$.*

(iii) *For $i = 1, 2, \ldots, r$, $\max\{|X_i|, |Y_i|\} \leq \kappa$, $|X_i| + |Y_i| \geq (4/3)\kappa$ and each of X_i and Y_i contains at least one of the lightest $(2/3)\kappa$ terminals among $X_i \cup Y_i$ in terms of vertex weight d.*

(iv)
$$\sum_{Z \in \mathcal{M}_1} w(T\langle Z \cup \{s\}\rangle) + \sum_{Z \in \mathcal{M}_2 \cup \mathcal{M}_3} w(T\langle Z\rangle) \leq (4/3)w(T).$$

Furthermore, such a partition $\mathcal{M}$ can be computed in polynomial time. □

To prove Lemma 2.7, we can assume without loss of generality that in a given tree T, (i) all terminals are leaves, i.e., $M = L(T)$, by introducing a new edge of weight zero for each non-leaf terminal, and (ii) $|Ch(v)| = 2$ holds for every non-leaf $v \in V(T)$, i.e., T is a binary tree rooted at s, by splitting vertices of degree more than 3 with new edges of weight zero.

We prove Lemma 2.7 by showing that the next algorithm actually delivers a desired partition $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$. The algorithm can be outlined as follows. Choose a vertex $v \notin Q \cup \{s\}$ with the maximum depth in the current tree such that $Z_v = D(v) \cap M$ contains at least $(2/3)\kappa$ terminals, where Q is initialized to be empty and is used throughout the algorithm to keep track of all vertices v chosen by the algorithm. Depending on the number of terminals in Z_v, we add Z_v to $\mathcal{M}_2$, add v to Q, or compute a partition of Z_v by using Lemma 2.3 or 2.5. In the latter case, we add the subsets in the obtained partition to one of $\mathcal{M}_2$ and $\mathcal{M}_3$. Figure 2.5 summarizes all possible cases of Z_v considered by the algorithm and the property and the action associated with each case. Remove any terminal in $\mathcal{M}_2 \cup \mathcal{M}_3$ from M. Repeat these steps on the minimal subtree of T that contains the current set M of the remaining terminals and s until $V(T) - (Q \cup \{s\})$ contains no more such vertex v in T. Finally, we partition the set of the remaining terminals by using Lemma 2.6, and add the obtained subsets to one of $\mathcal{M}_1$ or $\mathcal{M}_2$. A formal description of the algorithm is as follows.

Algorithm TreeCover
Input: A binary tree $\widehat{T}$ rooted at s, a set $M = L(\widehat{T})$ of terminals, a positive integer κ, an edge weight function $w : E(T) \to R^+$, and a vertex weight function $d : M \to R^+$.
Output: A partition $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$ of M that satisfies the conditions in Lemma 2.7.
1. Let $T := \widehat{T}$; $Q := \mathcal{M}_2 := \mathcal{M}_3 := \emptyset$;
2. **while** there exists a vertex $v \in V(T) - \{s\} - Q$ such that $|M \cap D_T(v)| \geq (2/3)\kappa$ **do**

Let $v \in V(T) - \{s\} - Q$ be the vertex with the maximum depth and $|D_T(v) \cap M| \geq (2/3)k$, where for each $u \in Ch_T(v)$, $|D_T(u) \cap M| < (2/3)k$ or $T\langle D_T(u) \cap M\rangle$ contains a 2/3-balance-tree and satisfies $|D_T(u) \cap M| < (4/3)k$

	property	action
Case-1: $\lvert Z_v\rvert \leq k$		$\mathcal{M}_2 := \mathcal{M}_2 \cup \{Z_v\}$;
Case-2: $k < \lvert Z_v\rvert < (4/3)k$	T_v contains a 2/3-balance-tree	$Q := Q \cup \{v\}$ (Z_v does not always have a partition satisfying conditions in Lemma 2.7. T_v will be a subtree of a tree handled either in a subsequent iteration or in Line 27 of the algorithm)
Case-3: $(4/3)k \leq \lvert Z_v\rvert \leq 2k$	one branch (B_1) of T_v contains a 2/3-balance-tree	Reroot T_v at the root of the balance-tree; Apply Lemma 2.3 to the resulting tree to obtain (i) $Z_v = \{X, Y\}$; $\mathcal{M}_2 := \mathcal{M}_2 \cup \{X, Y\}$, or (ii) $Z_v = \{X, Y\}$; $\mathcal{M}_3 := \mathcal{M}_3 \cup \{X, Y\}$
Case-4: $2k < \lvert Z_v\rvert < (8/3)k$	each branch of T_v contains a 2/3-balance-tree	Apply Lemma 2.5 to T_v to obtain (i) $Z_v = \{A, B, C\}$; $\mathcal{M}_2 := \mathcal{M}_2 \cup \{A, B, C\}$, or (ii) $Z_v = \{A, B, C\}$; $\mathcal{M}_2 := \mathcal{M}_2 \cup \{A\}$; $\mathcal{M}_3 := \mathcal{M}_3 \cup \{B, C\}$

$M := M - (\mathcal{M}_2 \cup \mathcal{M}_3)$; $T := T\langle M \cup \{s\}\rangle$

Figure 2.5: Illustration of one iteration of the while-loop in algorithm TreeCover.

3. Choose such v with the maximum depth from s;
4. Let $Z_v := M \cap D_T(v)$; $T_v := T\langle Z_v\rangle$;
5. Let B_1 and B_2 be the two branches of T_v;
6. Let $Z_i = M \cap V(B_i)$, $i = 1, 2$, where $|Z_1| \geq |Z_2|$;
7. **begin** /* Distinguish the next four cases. */
8. **Case-1** $|Z_v| \leq \kappa$: Let $\mathcal{M}_2 := \mathcal{M}_2 \cup \{Z_v\}$;
9. **Case-2** $\kappa < |Z_v| < (4/3)\kappa$: Let $Q := Q \cup \{v\}$;
10. **Case-3** $(4/3)\kappa \leq |Z_v| \leq 2\kappa$:
11. Let y be the root of the 2/3-balance-tree in B_1, and regard T_v as a tree T'_y rooted at y;
12. Apply Lemma 2.3 to T'_y to get a partition $\{X, Y\}$ of Z_v that satisfies one of conditions(i)-(ii) in Lemma 2.3;
13. **if** (i) holds **then** /* we get a partition $\{X, Y\}$ of Z_v*/
$\mathcal{M}_2 := \mathcal{M}_2 \cup \{X, Y\}$
14. **else** /* (ii) holds, i.e., we get a partition $\{X, Y\}$ of Z_v, where Z_0 consists of the lightest $(2/3)\kappa$ terminals in Z_v with respect to d */
$\mathcal{M}_3 := \mathcal{M}_3 \cup \{X, Y\}$;
15. **endif**

16. **Case-4** $2\kappa < |Z_v| < (8/3)\kappa$:
17. Apply Lemma 2.5 to T_v to get a partition of Z_v that satisfies one of conditions(i)-(ii) in Lemma 2.5;
18. **if** (i) holds **then** /* we get a partition $\{A, B, C\}$ of Z_v*/
    $\mathcal{M}_2 := \mathcal{M}_2 \cup \{A, B, C\}$
19. **else** /* (ii) holds, i.e., we get a partition $\{A, B, C\}$ of Z_v, where Z_0 consists of the lightest $(2/3)\kappa$ terminals in $\overline{A} = B \cup C$ with respect to d */
    $\mathcal{M}_2 := \mathcal{M}_2 \cup \{A\}$; $\mathcal{M}_3 := \mathcal{M}_3 \cup \{B, C\}$
20. **endif**
21. **end**; /* Cases-1,2,3,4 */
22. Let $M := M - (\mathcal{M}_2 \cup \mathcal{M}_3)$; $T := T\langle M \cup \{s\}\rangle$
23. **endwhile**;
24. **if** $M = \emptyset$ **then**
25. $\mathcal{M}_1 := \emptyset$
26. **else** /* $M \neq \emptyset$ */
27. Regard T as a tree T_s rooted at s and apply Lemma 2.6 to T_s to get a partition $\mathcal{Z}_1 \cup \mathcal{Z}_2$ of M that satisfies the conditions in Lemma 2.6;
28. $\mathcal{M}_1 := \mathcal{Z}_2$; $\mathcal{M}_2 := \mathcal{M}_2 \cup \mathcal{Z}_1$
29. **endif**.

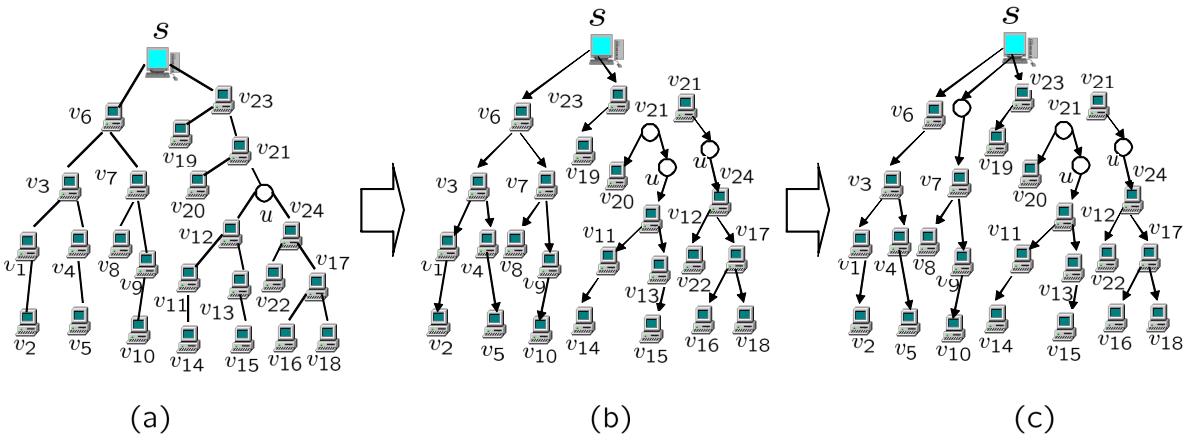

Figure 2.6: Illustration of algorithm TreeCover; (a) a minimum tree of the graph given in Fig. 2.1(a); (b) and (c) illustrate intermediate iterations of algorithm TreeCover applying to the tree in (a).

Figure 2.6 illustrates a computation process of this algorithm applied to a minimum tree of the graph given in Fig. 2.1(a) that spans source and all terminals. In Fig. 2.6(a), the algorithm adds vertex u to Q in the first iteration since T_u is a 2/3-balance-tree. In the second iteration (Case-3 holds), the algorithm regards the subtree of T rooted at vertex v_{21}

as a tree T' rooted at u and applies Lemma 2.3 to T' (see Fig. 2.1(b)). Finally, the algorithm adds vertex v_6 to Q (T_{v_6} is a 2/3-balance-tree) and then applies Lemma 2.6 to the tree on the source and the remaining terminals (see Fig. 2.1(c)).

Proof of Lemma 2.7. We will prove the correctness of algorithm TreeCover and then show that the partition obtained from this algorithm satisfies the conditions in Lemma 2.7. We first prove by induction the correctness of algorithm TreeCover. Let B_1 and B_2 (resp., Z_1 and Z_2) be as defined in the algorithm. We first consider the first iteration of the while-loop. By the choice of vertex v in line 3, $\max\{|Z_1|, |Z_2|\} < (2/3)\kappa$. Hence $(2/3)\kappa \leq |Z_v| < (4/3)\kappa$ holds in line 4, which implies that only Case-1 or Case-2 can occur in the first iteration. If $|Z_v| \leq \kappa$ holds, then Z_v is removed from M and added to $\mathcal{M}_2$ in Case-1. Otherwise ($\kappa < |Z_v| < (4/3)\kappa$) Case-2 holds, where the two branches B_1 and B_2 of the vertex v satisfy $(1/3)\kappa < |Z_1|, |Z_2| < (2/3)\kappa$ and $|Z_v| = |Z_1| + |Z_2| > \kappa$, and hence T_v is a 2/3-balance-tree. In the latter case, v is added to a set Q.

Assume that the algorithm works correctly after the execution of the jth iteration of the while-loop. We show the correctness of the algorithm during the execution of the $(j+1)$st iteration. Note that, for any vertex v chosen in line 3, set Z_v will be removed from the current set M except for Case-2. Now let v be a vertex selected in line 3 in the $(j+1)$st iteration. Then we see that, for each child $u \in Ch_T(v)$, either (i) $|M \cap D_T(u)| < (2/3)\kappa$ holds (if u has not been chosen in line 3) or (ii) $u \in Q$ holds and T_u contains a 2/3-balance-tree and satisfies $\kappa < |M \cap D_T(u)| < (4/3)\kappa$ (otherwise). Therefore, one of $(2/3)\kappa \leq |Z_v| \leq \kappa$, $\kappa < |Z_v| < (4/3)\kappa$, $(4/3)\kappa \leq |Z_v| \leq 2\kappa$, and $2\kappa < |Z_v| < (8/3)\kappa$ holds. Now if $(2/3)\kappa \leq |Z_v| \leq \kappa$ holds, then Z_v is removed from the current set M in Case-1 after it is added to $\mathcal{M}_2$. If $\kappa < |Z_v| < (4/3)\kappa$ (i.e., Case-2) holds, then T_v is a 2/3-balance-tree (if $(1/3)\kappa < |Z_1|, |Z_2| < (2/3)\kappa$) or B_1 (consequently T_v) contains a 2/3-balance-tree (by $|Z_1| \geq |Z_2|$). In this case, the vertex v is added to set Q. Analogously with Case-2, we see that, if $(4/3)\kappa \leq |Z_v| \leq 2\kappa$ (i.e., Case-3) holds then B_1 contains a 2/3-balance-tree, where $|Z_1| > \kappa$ and $|Z_2| < (2/3)\kappa$. Hence tree T'_y defined in line 11 satisfies the conditions of Lemma 2.3 and a desired partition of Z_v can be constructed in line 12. In the last case, where $2\kappa < |Z_v| < (8/3)\kappa$ (i.e., Case-4) holds, we can also observe that, for each $i = 1, 2$, B_i contains a 2/3-balance-tree and $\kappa < |Z_i| < (4/3)\kappa$ holds. This implies that tree T_v in line 17 satisfies the conditions of Lemma 2.5 and a desired partition of Z_v can be constructed in line 17. In Cases-3 and 4, Z_v is removed from the current set M after elements of its partition are added to appropriate subsets of $\mathcal{M}$. Therefore, the algorithm works correctly during the execution of all iterations of the while-loop.

After the final iteration of the while-loop, there is no vertex $v \in V(T) - \{s\} - Q$ such that $|M \cap D_T(v)| \geq (2/3)\kappa$. Hence, if the current set M is not empty, then for each child $u \in Ch_T(s)$, either (i) $|M \cap D_T(u)| < (2/3)\kappa$ holds (if u has not been chosen in line 3) or (ii) $u \in Q$ holds and T_u contains a 2/3-balance-tree and satisfies $\kappa < |M \cap D_T(u)| < (4/3)\kappa$ (otherwise). That is, tree T_s defined in line 27 satisfies the conditions in Lemma 2.6 and

a desired partition of the current M can be constructed in line 27. This completes the correctness of algorithm TreeCover.

Now we prove that the partition obtained from algorithm TreeCover satisfies conditions (i)-(iv) in Lemma 2.7. We observe that conditions (i), (ii), and (iii) in Lemma 2.7 follow immediately from construction of $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$. Finally we show (iv). By Lemma 2.3, a partition $\{X, Y\}$ of Z_v in line 12 satisfies $w(T\langle X\rangle) + w(T\langle Y\rangle) \leq (4/3)w(T_v)$. From Lemma 2.5, a partition $\{A, B, C\}$ of Z_v obtained in line 17 satisfies $w(T\langle A\rangle) + w(T\langle B\rangle) + w(T\langle C\rangle) \leq (4/3)w(T_v)$. Moreover, $\sum_{Z\in\mathcal{Z}_1} w(T\langle Z\rangle) + \sum_{Z\in\mathcal{Z}_2} w(T\langle Z \cup \{s\}\rangle) \leq (4/3)w(T_s)$ holds for the partition obtained in line 27. Hence by summing the latter inequality and the resultant inequalities over all iterations of the while-loop, we get (iv) in Lemma 2.7. This completes the proof of Lemma 2.7. □

2.4 Approximation algorithm

This section describes a framework of our approximation algorithm for the unit demand case of CMTR and then analyzes its approximation ratio. The algorithm relies on the results on tree covers in a tree provided in Section 2.3.

Consider CMTR instance given in Fig. 2.1(a). The algorithm first produces a tree of the minimum cost including all vertices in $M \cup \{s\}$ given in Fig. 2.6(a), partitions this tree into a set of subtrees given in Fig. 2.6(c), by applying algorithm TreeCover to this tree, and finally connects the closest terminal in each subtree to s to construct the approximate solution presented in Fig. 2.1(b). The entire algorithm is described as follows.

Algorithm UnitCMTR
Input: An instance $I = (G, w, \kappa, s, M)$ of CMTR.
Output: A solution $(\mathcal{M}, \mathcal{T})$ to I.

Step 1. Compute a minimum tree T that spans $M \cup \{s\}$ in G.

Step 2. Regard T as a tree rooted at s, and define $d : M \to R^+$ by setting $d(t)$ to be the distance from s to $t \in M$, i.e., the sum of weights (in term of w) of edges in a shortest path $SP(s, t)$ from s to t in G.
Apply Lemma 2.7 to (T, M, w, s, κ, d) to obtain a partition

$$\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$$

of M, where $\mathcal{M}_3 = \{X_1, Y_1, \ldots, X_r, Y_r\}$, that satisfies conditions (i)-(iv) of the lemma.

Step 3. For each terminal subset $Z \in \mathcal{M}_1$, let $T_Z := T\langle Z \cup \{s\}\rangle$.
For each terminal subset $Z \in \mathcal{M}_2 \cup \mathcal{M}_3$, choose a terminal $t_Z \in Z$ with the minimum distance $d(t_Z)$, and let T_Z be the tree obtained from $T\langle Z\rangle$ by adding the edge set of a shortest path $SP(s, t_Z)$ from s to t_Z in G.

Step 4. Let $\mathcal{T} = \{T_Z \mid Z \in \mathcal{M}\}$, and output $(\mathcal{M}, \mathcal{T})$. □

We show that algorithm UnitCMTR has the following performance.

Theorem 2.2. *For an instance $I = (G, w, \kappa, s, M)$ of* CMTR*, algorithm* UnitCMTR *delivers a $(3/2 + (4/3)\rho_{\mathrm{ST}})$-approximate solution $(\mathcal{M}, \mathcal{T})$, where ρ_{ST} is the ratio of $w(T)$ to the minimum cost of a Steiner tree that spans $M \cup \{s\}$.*

Proof. We first show that algorithm UnitCMTR produces a feasible solution. By conditions (i)-(iii) of Lemma 2.7, every subset $Z \in \mathcal{M}$ consists of at most κ terminals, and thereby a solution $(\mathcal{M}, \mathcal{T})$ obtained by algorithm UnitCMTR is feasible to I.

We next show that $(\mathcal{M}, \mathcal{T})$ is a $(3/2 + (4/3)\rho_{\mathrm{ST}})$-approximate solution. Let $opt(I)$ denote the weight of an optimal solution. By construction, the cost of $\mathcal{T}$ is bounded by

$$\sum_{T' \in \mathcal{T}} w(T') \leq \sum_{Z \in \mathcal{M}_1} w(T\langle Z \cup \{s\}\rangle) + \sum_{Z \in \mathcal{M}_2 \cup \mathcal{M}_3} w(T\langle Z\rangle) + \sum_{Z \in \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z),$$

which is at most

$$(4/3)w(T) + \sum_{Z \in \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z)$$

by condition (iv) of Lemma 2.7.

For a minimum Steiner tree T^* that spans $M \cup \{s\}$, we have $w(T) \leq \rho_{\mathrm{ST}} w(T^*)$ and $w(T^*) \leq opt(I)$ by Lemma 2.1. Hence $(4/3)w(T) \leq (4/3)\rho_{\mathrm{ST}} \cdot opt(I)$ holds. To prove the theorem, it suffices to show that

$$\sum_{Z \in \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z) \leq (3/2) opt(I). \tag{2.6}$$

Consider a subset $Z \in \mathcal{M}_2$. By the choice of terminal $t_Z \in Z$ and condition (ii) of Lemma 2.7, we have

$$\sum_{t \in Z} d(t) \geq |Z| d(t_Z) \geq (2/3)\kappa d(t_Z). \tag{2.7}$$

Now consider a pair of subsets $X_i, Y_i \in \mathcal{M}_3$, and their terminals $t_Z = t_{X_i} \in X_i$ and $t_Z = t_{Y_i} \in Y_i$ chosen in Step 3. Assume without loss of generality that $d(t_{X_i}) \leq d(t_{Y_i})$. Then t_{X_i} has the smallest distance among all terminals in $X_i \cup Y_i$. Hence for the set $Z_0 \subseteq X_i \cup Y_i$ of terminals with the first $(2/3)\kappa$ smallest distance, we have

$$\sum_{t \in Z_0} d(t) \geq (2/3)\kappa \cdot d(t_{X_i}).$$

For the set $(X_i \cup Y_i) - Z_0$ of the remaining terminals, we have

$$\sum_{t \in (X_i \cup Y_i) - Z_0} d(t) \geq \big(|X_i| + |Y_i| - (2/3)\kappa\big) d(t_{Y_i}) \geq (2/3)\kappa \cdot d(t_{Y_i}),$$

where the last inequality follows from $|X_i| + |Y_i| \geq (4/3)\kappa$ in condition (iii) of Lemma 2.7. Therefore, it holds

$$\sum_{t \in (X_i \cup Y_i)} d(t) \geq (2/3)\kappa \cdot d(t_{X_i}) + (2/3)\kappa \cdot d(t_{Y_i}). \tag{2.8}$$

By summing inequalities (2.7) and (2.8) overall subsets in $\mathcal{M}_2 \cup \mathcal{M}_3$, we have

$$(2/3)\kappa \sum_{Z \in \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z) \leq \sum_{t \in Z \in \mathcal{M}_2 \cup \mathcal{M}_3} d(t) \leq \sum_{t \in M} d(t). \tag{2.9}$$

By Lemma 2.1, this implies

$$\sum_{Z \in \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z) \leq (3/2)(1/\kappa) \sum_{t \in M} d(t) \leq (3/2) opt(I),$$

from which (2.6) follows. □

Chapter 3

Multicast Routing Problem in a Network with Multi-sources

In this chapter we extend CMTR problem described in the previous chapter to the case of multisources. That is, instead of designating a single source in the network, we are given a source set $S \subseteq V$ with a weight $g(s) \geq 0$, $s \in S$, and the problem asks to find a subset $S' \subseteq S$, a partition $\{Z_i\}$ of a terminal set M, and a set of trees T_i of a given graph G such that, for each i, $q(Z_i) \leq \kappa$ and T_i spans $Z_i \cup \{s\}$ for some $s \in S'$.

3.1 Introduction

In an interesting generalization of the multicast routing problem, a group of vertices of the underlying network is designated as sources such that each source is associated with an opening cost. In this case, the problem of finding a multicast routing is to open a set of sources and construct a set of multicast trees such that each of these trees spans an opened source and the terminals selected to receive information from this source. The objective of this problem is to minimize the cost of constructing the multicast trees plus the cost for opening the sources. In this chapter, we study such a multi-source version of CMTR. The problem, called the *capacitated multi-source multicast tree routing problem* (CMMTR for short), can be formally stated as follows.

Capacitated Multi-source Multicast Tree Routing Problem (CMMTR):
Input: A graph $G = (V, E)$, an edge weight function $w : E \rightarrow R^+$, an upper limit $\kappa > 0$ on the total demand in a multicast tree, a set $S \subseteq V$ of sources, an opening cost function $g : S \rightarrow R^+$, a set $M \subseteq V - S$ of terminals, and a demand function $q : M \rightarrow R^+$.
Feasible solution: A subset $S' \subseteq S$, a partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_\ell\}$ of M, and a set $\mathcal{T} = \{T_1, T_2, \ldots, T_\ell\}$ of trees of G such that, for each $i = 1, 2, \ldots, \ell$,

$q(Z_i) \leq \kappa$ holds, and

Table 3.1: Approximation algorithms for CCFL and CMMTR problems

Problem	**CCFL**	
	unit demands $q \equiv 1$	general demands $q \geq 0$
Single sink	$1 + \rho_{\mathrm{ST}}$ [33]	$2 + \rho_{\mathrm{ST}}$ [33]
Multi-sink	$\rho_{\mathrm{UFL}} + \rho_{\mathrm{ST}}$ [61]	$2\rho_{\mathrm{UFL}} + \rho_{\mathrm{ST}}$ [61]
Problem	**CMMTR**	
	unit demands $q \equiv 1$	general demands $q \geq 0$
Single source	$8/5 + (5/4)\rho_{\mathrm{ST}}$ [11], $3/2 + (4/3)\rho_{\mathrm{ST}}$ [56] (Chapter 2)	$2 + \rho_{\mathrm{ST}}$ [40] (Chapter 2)
Multi-source	$(3/2)\rho_{\mathrm{UFL}} + (4/3)\rho_{\mathrm{ST}}$ [55] [this chapter]	$2\rho_{\mathrm{UFL}} + \rho_{\mathrm{ST}}$ [55] [this chapter]

$$T_i \text{ contains } Z_i \cup \{s\} \text{ for some } s \in S',$$

where two or more subtrees of $\mathcal{T}$ can contain a common source $s \in S'$.
Goal: Minimize

$$\sum_{T_i \in \mathcal{T}} w(T_i) + \sum_{s \in S'} g(s).$$

CMTR is CMMTR with $|S| = 1$, and is known to be NP-hard. If $q(v) = 1$ for all $v \in M$ and κ is a positive integer in an instance of CMMTR, then we call the problem of such instances the *unit demand case* of CMMTR, and its instance may be written as (G, w, κ, S, g, M).

CMMTR is closely related to the *capacitated-cable facility location problem* (CCFL for short). An instance $(G, w, \lambda, S, g, M, q)$ of CCFL consists of an undirected graph $G = (V, E)$, an edge weight function $w : E \to R^+$, a capacity $\lambda > 0$ of each edge, a set S of sinks, an opening cost function $g : S \to R^+$, a set $M \subseteq V - S$ of clients, and a demand function $q : M \to R^+$. Then for each client $v \in M$, we want to send its demand $q(v)$ along a single path P_v from v to an opened sink in S in G (demand $q(v)$ cannot be split). A set of such paths can pass through an edge in G as long as the total demand of the paths does not exceed the capacity λ. For any edge $e \in E$, we are allowed to install any integer number $h(e)$ of copies of e, if necessary. A pair of a subset $S' \subseteq S$ and a set $\{h(e) \mid e \in E\}$ of integers is a feasible solution to CCFL if there is a set $\{P_v \subseteq E' \mid v \in M\}$ of paths, each of which connects v and a sink $s \in S'$, such that, for each edge $e \in E$, it holds $\sum_{v:e \in E(P_v)} q(v) \leq h(e)\lambda$, i.e., the total demand of the paths P_v passing through e is no more than $h(e)\lambda$. Then CCFL asks to

find a feasible pair (S', E') that minimizes the sum of the cost for opening S' and the cost for installing edges in E, i.e., $\sum_{s \in S'} g(s) + \sum_{e \in E} h(e)w(e)$.

Ravi and Sinha [61] gave a $(\rho_{\mathrm{UFL}} + \rho_{\mathrm{ST}})$-approximation algorithm for CCFL problem with the unit demands and a $(2\rho_{\mathrm{UFL}} + \rho_{\mathrm{ST}})$-approximation algorithm for that with the general demands, where ρ_{UFL} is any approximation ratio achievable for UFL in a metric (G, w) (see Section 1.5 for the definition and known algorithms of UFL).

In this chapter, we propose a $(2\rho_{\mathrm{UFL}} + \rho_{\mathrm{ST}})$-approximation algorithm to CMMTR and a $((3/2)\rho_{\mathrm{UFL}} + (4/3)\rho_{\mathrm{ST}})$-approximation algorithm to the unit demand case of CMMTR. As in the approximation algorithms to CCFL due to Ravi and Sinha [61], we also use an approximation result on the metric UFL to derive our approximation algorithms to CMMTR.

Table 6.1 shows a summary of the approximation algorithms for CCFL and CMMTR.

3.2 Preliminaries

We now introduce two lower bounds on the optimal value of CMMTR. The first lower bound is based on the Steiner tree problem.

Lemma 3.1. *Given a CMMTR instance $I = (G, w, \kappa, S, g, M, q)$, let $G' = (V \cup \{r\}, E \cup E')$ be the graph obtained by introducing a new vertex $r \notin V$ and a set of new edges $E' = \{(r, s) \mid s \in S\}$ with weight $w(r, s) = g(s)$. Then the minimum cost of a Steiner tree to $(G', w, M \cup \{r\})$ is a lower bound on the optimal value to CMMTR instance I.*

Proof. Consider an optimal solution $(S', \mathcal{M}, \mathcal{T})$ to CMMTR instance I. Let $E(\mathcal{T}) = \cup_{T_i \in \mathcal{T}} E(T_i)$ $(\subseteq E)$, i.e., the set of all edges used in the optimal solution. Then the edge set $E(\mathcal{T}) \cup \{(r, s) \mid s \in S'\}$ contains a tree T that spans $M \cup \{r\}$ in G'. We see that the cost $w(T)$ of T in G' is at most that of CMMTR solution. Hence the minimum cost of a Steiner tree to $(G', w, M \cup \{r\})$ is no more than the optimal value to CMMTR instance I. □

To introduce the second lower bound, we recall that an instance (H, c, F, f, C, b) of UFL consists of an undirected graph H, an edge weight function $c : E(H) \to R^+$, a set F of facilities, an opening cost function $f : F \to R^+$, a set $C = V(H) - F$ of clients, and a demand function $b : C \to R^+$, where $f(i)$ means the cost of opening facility i, and $c(i, j)$ means the cost for connecting facility $i \in F$ and client $j \in C$. The goal is to identify a subset $F' \subseteq F$ of facilities that minimizes

$$\Phi(F') := \sum_{i \in F'} f(i) + \sum_{j \in C} b(j) \min_{i \in F'} c(i, j).$$

Lemma 3.2. *Given a CMMTR instance $I = (G, w, \kappa, S, g, M, q)$, let $I' = (H, c, F, f, C, b)$ be a UFL instance obtained by setting $F := S$, $C := M$, $f := g$, $b := q$, $H := (F \cup C, \binom{F \cup C}{2})$, and*

$$c(u, v) := d_{(G,w)}(u, v)/\kappa, \quad u, v \in F \cup C.$$

Then, (H, c) is metric, and the cost $\Phi(F')$ of an optimal solution $F' \subseteq F$ to UFL instance I' is a lower bound on the optimal value to CMMTR instance I.

Proof. It is easy to see that (H, c) is metric. Consider an optimal solution $(S', \mathcal{M}, \mathcal{T})$ to CMMTR instance I, where we denote by $s_i \in S'$ the source in $V(T_i) \cap S'$. Regard $F' := S'$ as a solution to UFL instance I', and consider its cost $\Phi(F')$. For each $v \in C = M$, let s_v be the source $s_i \in V(T_i)$ of the tree $T_i \in \mathcal{T}$ with $v \in Z_i \in \mathcal{M}$. Then $\Phi(F') \leq \sum_{s \in F'} f(s) + \sum_{v \in C} b(v)c(s_v, v)$ holds. On the other hand, for each tree $T_i \in \mathcal{T}$, we see that $\sum_{v \in Z_i} b(v)w(s_i, v) \leq \sum_{v \in Z_i} b(v)d_{(T,w)}(s_i, v)/\kappa$ since $\sum_{v \in Z_i} b(v) \leq \kappa$, and hence $\sum_{v \in Z_i} b(v)d_{(G,w)}(s_i, v)/\kappa \leq w(T_i)$ holds. Therefore, we have $\Phi(F') \leq \sum_{s \in S'} g(s) + \sum_{T_i \in \mathcal{T}} w(T_i)$, which implies that the cost of an optimal solution to UFL instance I' is no more than that of an optimal solution to CMMTR instance I, as required. □

The above two lower bounds are used in the algorithm for CCFL [61]. In this chapter, we give two approximation algorithms to CMMTR instances by using Lemmas 3.1 and 3.2 and our new results on tree covers introduced in Chapter 2.

3.3 General demands

This section presents our approximation algorithm for CMMTR with the general demands. The algorithm relies on a κ-balanced partition of a set of terminals in a tree (see Section 2.2).

For convenience, we recall the definition of κ-balanced partition. For a tree T rooted at a vertex r, an ordered partition $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_p\}$ of a subset of the terminal set M is called *κ-balanced* if the following holds:

(i) $q(Z_i) \leq \kappa$ for $i = 1, 2, \ldots, p$;

(ii) $q(Z_i) > \kappa/2$ for $i = 1, 2, \ldots, p-1$, and if $p \geq 2$ then $q(Z_{p-1} \cup Z_p) > \kappa$; and

(iii) Each $T\langle Z_j \rangle$ $(j = 1, 2, \ldots, p-1)$ has no common edge with $T\langle \cup_{j<i\leq p} Z_i + r \rangle$.

Based on κ-balanced partition, we obtain the following approximation algorithm for CMMTR with general demands, where we first choose a subset S_1 of sources by solving an UFL instance, compute a Steiner tree T in an augmented graph G' and finally construct a partition $\mathcal{M}$ of a given terminal set M and a set $\mathcal{T}$ of trees based on S_1 and T.

Algorithm GeneralCMMTR
Input: An instance $I = (G, w, \kappa, S, g, M, q)$ of CMMTR.
Output: A solution $(S', \mathcal{M}, \mathcal{T})$ to I.

Step 1. Construct UFL instance $I' = (H, c = d_{(G,w)}/\kappa, F = S, f = g, C = M, b = q)$ defined in Lemma 3.2. Find a ρ_{UFL}-approximate solution $S_1 \subseteq F = S$ to UFL instance I' (see Fig. 3.1(a)).

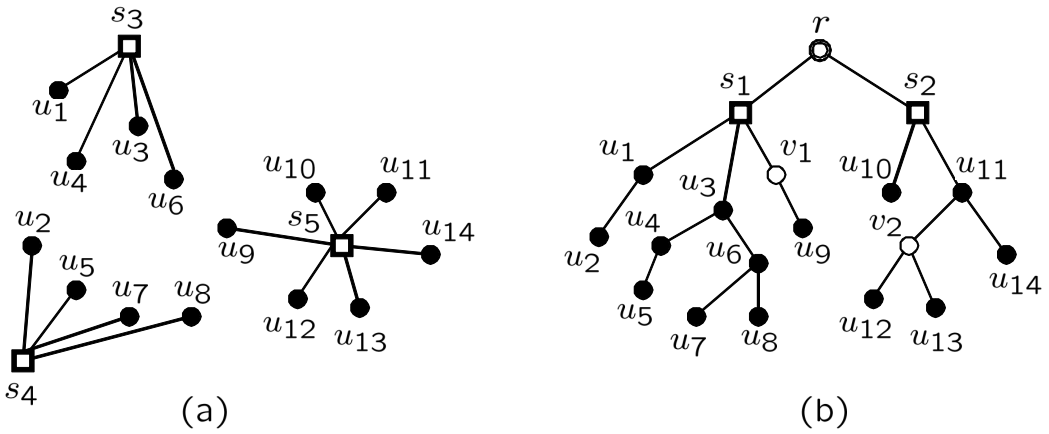

Figure 3.1: Illustration for algorithm GeneralCMMTR applying to an instance of CMMTR with source set $S = \{s_1, \ldots, s_6\}$ and terminal set $M = \{u_1, \ldots, u_{13}\}$; (a) a ρ_{UFL}-approximate solution to UFL instance I' defined in Steps 1; (b) a ρ_{ST}-approximate solution to the Steiner tree problem to $(G', w, M \cup \{r\})$ defined in Step 2.

Step 2. Let r be a new vertex (i.e., $r \notin V$), and construct the edge-weighted graph $G' = (V \cup \{r\}, E \cup E')$, where $E' = \{(r, s) \mid s \in S\}$ and $w(r, s) = g(s)$, $s \in S$. Let $(G', w, M \cup \{r\})$ be an instance of the Steiner tree problem. Let tree T be a ρ_{ST}-approximate solution to $(G', w, M \cup \{r\})$. Let $S_2 := S \cap V(T)$ (see Fig. 3.1(b)).

Step 3. Regard T as a tree rooted at r. Define a function $d : M \to R^+$ by setting

$$d(t) := \min_{s \in S_1} d_{(G,w)}(s, t), \quad t \in M, \tag{3.1}$$

and let $\sigma(t)$ denote a source $s \in S_1$ with $d(t) = d_{(G,w)}(s, t)$.
For each $s \in S_2$,

let T_s be the subtree of T rooted at s, and find a κ-balanced partition

$$\mathcal{M}^{(s)} = \{Z_1^{(s)}, Z_2^{(s)}, \ldots, Z_{p_s}^{(s)}\}$$

of $M \cap V(T_s)$ in T_s that satisfies conditions (i)-(iii).
Let $T_{Z_{p_s}^{(s)}} := T\langle Z_{p_s}^{(s)} \cup \{s\}\rangle$.

Step 4. Let $\mathcal{M}_1 := \cup_{s \in S_2} Z_{p_s}^{(s)}$ and $\mathcal{M}_2 := \cup_{s \in S_2} \mathcal{M}^{(s)} - \mathcal{M}_1$.
For each $Z \in \mathcal{M}_2$, choose a terminal $t_Z \in Z$ with the minimum weight $d(t_Z)$, and let T_Z be the tree obtained from $T\langle Z\rangle$ by adding a shortest path $SP(\sigma(t_Z), t_Z)$ between $\sigma(t_Z)$ and t_Z in (G, w).

Step 5. Let $\mathcal{M} := \mathcal{M}_1 \cup \mathcal{M}_2$, $\mathcal{T} := \{T_Z \mid Z \in \mathcal{M}\}$ and output $(S' = S_1 \cup S_2, \mathcal{M}, \mathcal{T})$ (see Fig. 3.2).

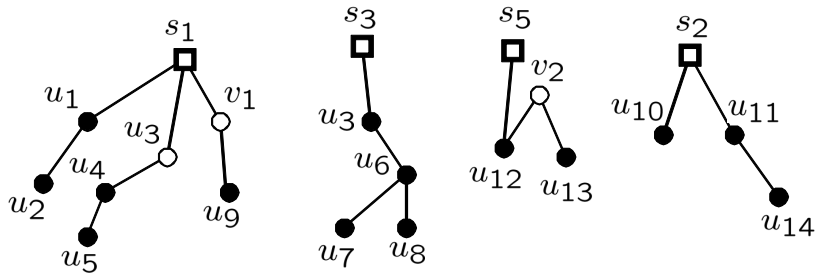

Figure 3.2: Output of algorithm GeneralCMMTR applied to example given in Fig. 3.1.

Figs. 3.1 and 3.2 illustrate a computation process of algorithm GeneralCMMTR to an instance $I = (G, w, \kappa = 5, S, g, M = \{u_1, u_2, \ldots, u_{14}\}, \{q(u_i) = 1 \mid i \leq 9\} \cup \{q(u_i) = 2 \mid 10 \leq i \leq 14\})$, where not all sources in S and edges in $E(G)$ are depicted. In Fig. 3.1(a), a ρ_{UFL}-approximate solution $S_1 = \{s_3, s_4, s_5\}$ to UFL instance I' is computed in Step 1. In Fig. 3.1(b), a ρ_{ST}-approximate solution T to the Steiner tree instance $(G', w, M \cup \{r\})$ is computed in Step 2, where $S_2 = \{s_1, s_2\}$. Fig. 3.2 describes a final solution to CMMTR instance I, where $S' = S_1 \cup S_2$, $\mathcal{M} = \{Z_1 = \{u_1, u_2, u_4, u_5, u_9\}, Z_2 = \{u_3, u_6, u_7, u_8\}, Z_3 = \{u_{12}, u_{13}\}, Z_4 = \{u_{10}\}, Z_5 = \{u_{11}, u_{14}\}\}$, and the set of the corresponding trees shown in the figure forms $\mathcal{T}$.

Theorem 3.1. *For an instance $I = (G, w, \kappa, S, g, M, q)$ of* CMMTR*, algorithm* General-CMMTR *delivers a $(2\rho_{\text{UFL}}+\rho_{\text{ST}})$-approximate solution $(S', \mathcal{M}, \mathcal{T})$, where ρ_{UFL} and ρ_{ST} are the approximation ratios of solutions S_1 and T to UFL and Steiner tree problems, respectively.*

Proof. Let $(S' = S_1 \cup S_2, \mathcal{M}, \mathcal{T})$ be a solution output by algorithm GeneralCMMTR. By condition (i) of κ-balanced partition, $q(Z) \leq \kappa$ for every subset $Z \in \mathcal{M}$. Moreover, the corresponding subtree $T_Z \in \mathcal{T}$ contains at least one source in $S_1 \cup S_2$. Hence the solution is feasible to I. We then show that it is a $(2\rho_{\text{UFL}} + \rho_{\text{ST}})$-approximate solution. Note that the total cost of the solution is

$$\sum_{s \in S_1 \cup S_2} g(s) + \sum_{T' \in \mathcal{T}} w(T').$$

Let $opt(I)$ denote the weight of an optimal solution to I. Note that the solution S_1 to UFL instance I' has cost

$$\begin{aligned} \Phi(S_1) &= \sum_{s \in S_1} g(s) + \sum_{t \in M} q(t) \min_{s \in S_1} d_{(G,w)}(s, t)/\kappa \\ &= \sum_{s \in S_1} g(s) + \sum_{t \in M} q(t) d(t)/\kappa, \end{aligned}$$

where d is the vertex weight function defined in (3.1). Then S_1 is a ρ_{UFL}-approximate solution to I', and we have by Lemma 3.2

$$\sum_{s \in S_1} g(s) + \sum_{t \in M} q(t) d(t)/\kappa \leq \rho_{\text{UFL}} \cdot opt(I). \tag{3.2}$$

Since tree T computed in Step 2 is a ρ_{ST}-approximate solution to $(G', w, M \cup \{r\})$, we have by Lemma 3.1

$$w(T) \leq \rho_{\mathrm{ST}} \cdot opt(I). \tag{3.3}$$

By construction, the cost of $\mathcal{T}$ is bounded by

$$\sum_{T' \in \mathcal{T}} w(T') \leq \sum_{s \in S_2} w(T\langle Z_{p_s}^{(s)} \cup \{s\}\rangle) + \sum_{Z \in \mathcal{M}_2} w(T\langle Z\rangle) + \sum_{Z \in \mathcal{M}_2} d(t_Z),$$

which is at most

$$\sum_{s \in S_2} w(T_s) + \sum_{Z \in \mathcal{M}_2} d(t_Z)$$

by condition (iii) of κ-balanced partition. Note that

$$\sum_{s \in S_2} w(T_s) \leq w(T) - \sum_{s \in S_2} w(r, s) = w(T) - \sum_{s \in S_2} g(s).$$

Hence it holds

$$\begin{aligned}
\sum_{s \in S_1 \cup S_2} g(s) + \sum_{T' \in \mathcal{T}} w(T') &\leq \sum_{s \in S_1 \cup S_2} g(s) + w(T) - \sum_{s \in S_2} g(s) + \sum_{Z \in \mathcal{M}_2} d(t_Z) \\
&\leq w(T) + \sum_{s \in S_1} g(s) + \sum_{Z \in \mathcal{M}_2} d(t_Z) \\
&\leq \rho_{\mathrm{ST}} \cdot opt(I) + \sum_{s \in S_1} g(s) + \sum_{Z \in \mathcal{M}_2} d(t_Z),
\end{aligned}$$

where the last inequality follows from (3.3).

Therefore, to prove the theorem, it suffices to show that

$$\sum_{s \in S_1} g(s) + \sum_{Z \in \mathcal{M}_2} d(t_Z) \leq 2\rho_{\mathrm{UFL}} \cdot opt(I). \tag{3.4}$$

For this, consider an arbitrary set $Z \in \mathcal{M}_2$. By the choice of terminal $t_Z \in Z$ and condition (ii) of κ-balanced partition, we have

$$\sum_{t \in Z} q(t)d(t) \geq d(t_Z) \sum_{t \in Z} q(t) > (\kappa/2)d(t_Z). \tag{3.5}$$

By summing inequality (3.5) overall subsets in $Z \in \mathcal{M}_2$, we have

$$(\kappa/2) \sum_{Z \in \mathcal{M}_2} d(t_Z) < \sum_{t \in Z \in \mathcal{M}_2} q(t)d(t) \leq \sum_{t \in M} q(t)d(t).$$

By (3.2), this implies

$$\begin{aligned}
\sum_{s \in S_1} g(s) + \sum_{Z \in \mathcal{M}_2} d(t_Z) &\leq \sum_{s \in S_1} g(s) + 2 \sum_{t \in M} q(t)d(t)/\kappa \\
&\leq 2\rho_{\mathrm{UFL}} \cdot opt(I),
\end{aligned}$$

proving (3.4), as required.

□

3.4 Unit demands

In this section, we handle the unit demand case of CMMTR, where κ is a positive integer representing an upper bound on the number of terminals in each multicast tree. We improve the approximation ratio $(2\rho_{\mathrm{UFL}} + \rho_{\mathrm{ST}})$ on the CMMTR with the general demands by relying on the result on tree covers proved in Lemma 2.7. For simplicity, we first recall Lemma 2.7.

Lemma 3.3. *Given a tree T rooted at a vertex r, an edge weight function $w : E(T) \to R^+$, a positive integer κ, a terminal set $M \subseteq V(T)$, and a vertex weight function $d : M \to R^+$, there is a partition $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \{X_1, Y_1, \ldots, X_p, Y_p\}$ of M that satisfies:*

(i) $|Z| < (2/3)\kappa$ *for all* $Z \in \mathcal{M}_1$.

(ii) $(2/3)\kappa \leq |Z| \leq \kappa$ *for all* $Z \in \mathcal{M}_2$.

(iii) *For each* $i = 1, 2, \ldots, p$, $\max\{|X_i|, |Y_i|\} \leq \kappa$, $|X_i| + |Y_i| \geq (4/3)\kappa$ *and each of* X_i *and* Y_i *contains the* $(2/3)\kappa$*-th lightest terminal or a lighter terminal in* $X_i \cup Y_i$ *in terms of vertex weight* d.

(iv)

$$\sum_{Z \in \mathcal{M}_1} w(T\langle Z \cup \{r\}\rangle) + \sum_{Z \in \mathcal{M}_2} w(T\langle Z\rangle) + \sum_{1 \leq i \leq p} (w(T\langle X_i\rangle) + w(T\langle Y_i\rangle)) \leq (4/3)w(T).$$

□

Based on Lemma 3.3, we obtain the following approximation algorithm for CMMTR, where, analogously with algorithm GeneralCMMTR, we first choose a subset S_1 of sources by solving a UFL instance, compute an approximate Steiner tree T in an augmented graph G' and finally construct a partition $\mathcal{M}$ of a given terminal set M and a set $\mathcal{T}$ of trees based on S_1 and T.

Algorithm UnitCMMTR
Input: An instance $I = (G, w, \kappa, S, g, M)$ of CMMTR.
Output: A solution $(S', \mathcal{M}, \mathcal{T})$ to I.

Step 1. Construct UFL instance $I' = (H, c = d_{(G,w)}/\kappa, F = S, f = g, C = M, b)$ defined in Lemma 3.2, where $b(v) = 1$, $v \in C$. Find a ρ_{UFL}-approximate solution $S_1 \subseteq F = S$ to the UFL instance I'.

Step 2. Construct the edge-weighted graph $G' = (V \cup \{r\}, E \cup E')$, where $E' = \{(r, s) \mid s \in S\}$ and $w(r, s) = g(s)$, $s \in S$, and let $(G', w, \{r\} \cup M)$ be an instance of the Steiner tree problem. Let tree T be a ρ_{ST}-approximate solution to $(G', w, \{r\} \cup M)$. Let $S_2 := S \cap V(T)$.

Step 3. Regard T as a tree rooted at r. Let d and σ be defined as in algorithm GENERAL-CMMTR.
For each $s \in S_2$,

let T_s be the subtree of T rooted at s, and apply Lemma 3.3 to $(T_s, w, \kappa, M \cap V(T_s), d)$ to obtain a partition

$$\mathcal{M}^{(s)} = \mathcal{M}_1^{(s)} \cup \mathcal{M}_2^{(s)} \cup \{X_i^{(s)}, Y_i^{(s)} \mid i = 1, 2, \ldots, p_s\}$$

of $M \cap V(T_s)$ that satisfies conditions (i)-(iv) of the lemma.
For each $Z \in \mathcal{M}_1^{(s)}$, let $T_Z := T\langle Z \cup \{s\}\rangle$.

Step 4. Let $\mathcal{M}_1 := \cup_{s \in S_2} \mathcal{M}_1^{(s)}$, $\mathcal{M}_2 := \cup_{s \in S_2} \mathcal{M}_2^{(s)}$, and $\{X_i, Y_i \mid i = 1, 2, \ldots, p\} := \cup_{s \in S_2} \{X_i^{(s)}, Y_i^{(s)} \mid i = 1, 2, \ldots, p_s\}$.
For each $Z \in \mathcal{M}_2 \cup \{X_i, Y_i \mid i = 1, 2, \ldots, p\}$, choose a terminal $t_Z \in Z$ with the minimum weight $d(t_Z)$, and let T_Z (resp., T_i' and T_i''), where $Z \in \mathcal{M}_2$ (resp., $Z \in \{X_i, Y_i\}$), be the tree obtained from $T\langle Z\rangle$ by adding a shortest path $SP(\sigma(t_Z), t_Z)$ between $\sigma(t_Z)$ and t_Z in (G, w).

Step 5. Let $\mathcal{T} := \{T_Z \mid Z \in \mathcal{M}_1 \cup \mathcal{M}_2\} \cup \{T_i', T_i'' \mid i = 1, 2, \ldots, p\}$.
Let $\mathcal{M} := \cup_{s \in S_2} \mathcal{M}^{(s)}$ and output $(S' = S_1 \cup S_2, \mathcal{M}, \mathcal{T})$. □

Theorem 3.2. *For an instance $I = (G, w, \kappa, S, g, M)$ of* CMMTR*, algorithm* UNITCMMTR *delivers a $((3/2)\rho_{\mathrm{UFL}} + (4/3)\rho_{\mathrm{ST}})$-approximate solution $(S', \mathcal{M}, \mathcal{T})$, where ρ_{UFL} and ρ_{ST} are the approximation ratios of solutions S_1 and T to UFL and Steiner tree problems, respectively.*

Proof. Let $(S' = S_1 \cup S_2, \mathcal{M}, \mathcal{T})$ be a solution output by algorithm UNITCMMTR. By conditions (i)-(iii) of Lemma 3.3, every subset $Z \in \mathcal{M}$ consists of at most κ terminals, and is contained the corresponding subtree $T_Z \in \mathcal{T}$ that contains at least one source in $S_1 \cup S_2$. Hence the solution is feasible to I. We then show that it is a $((3/2)\rho_{\mathrm{UFL}} + (4/3)\rho_{\mathrm{ST}})$-approximate solution. Note that the total cost of the solution is

$$\sum_{s \in S_1 \cup S_2} g(s) + \sum_{T' \in \mathcal{T}} w(T').$$

Let $opt(I)$ denote the weight of an optimal solution to I. Note that the solution S_1 to UFL instance I' has cost

$$\begin{aligned} \Phi(S_1) &= \sum_{s \in S_1} g(s) + \sum_{t \in M} \min_{s \in S_1} d_{(G,w)}(s, t)/\kappa \\ &= \sum_{s \in S_1} g(s) + \sum_{t \in M} d(t)/\kappa, \end{aligned}$$

where d is the vertex weight function defined in (3.1). Hence S_1 is a ρ_{UFL}-approximate solution to I', and we have by Lemma 3.2

$$\sum_{s\in S_1} g(s) + \sum_{t\in M} d(t)/\kappa \le \rho_{\mathrm{UFL}} \cdot opt(I). \tag{3.6}$$

Since tree T computed in Step 2 is a ρ_{ST}-approximate solution to $(G', \{r\} \cup M)$, we have by Lemma 3.1

$$w(T) \le \rho_{\mathrm{ST}} \cdot opt(I). \tag{3.7}$$

Let $\mathcal{Z} = \mathcal{M}_2 \cup \{X_i, Y_i \mid i = 1, 2, \ldots, p\}$. By construction, the cost of $\mathcal{T}$ is bounded by

$$\begin{aligned}\sum_{T'\in\mathcal{T}} w(T') \;\le\; & \sum_{s\in S_2} \sum_{Z\in\mathcal{M}_1^{(s)}} w(T\langle Z\cup\{s\}\rangle) + \sum_{1\le i\le p} (w(T\langle X_i\rangle) + w(T\langle Y_i\rangle)) \\ & + \sum_{Z\in\mathcal{M}_2} w(T\langle Z\rangle) + \sum_{Z\in\mathcal{Z}} d(t_Z),\end{aligned}$$

which is at most

$$(4/3)\sum_{s\in S_2} w(T_s) + \sum_{Z\in\mathcal{Z}} d(t_Z)$$

by condition (iv) of Lemma 3.3. Note that

$$\begin{aligned}(4/3)\sum_{s\in S_2} w(T_s) &\le (4/3)\big(w(T) - \sum_{s\in S_2} w(r,s)\big) \\ &= (4/3)\big(w(T) - \sum_{s\in S_2} g(s)\big).\end{aligned}$$

Hence it holds

$$\begin{aligned}\sum_{s\in S_1\cup S_2} g(s) + \sum_{T'\in\mathcal{T}} w(T') &\le \sum_{s\in S_1\cup S_2} g(s) + (4/3)\big(w(T) - \sum_{s\in S_2} g(s)\big) + \sum_{Z\in\mathcal{Z}} d(t_Z) \\ &\le (4/3)w(T) + \sum_{s\in S_1} g(s) + \sum_{Z\in\mathcal{Z}} d(t_Z) \\ &\le (4/3)\rho_{\mathrm{ST}} \cdot opt(I) + \sum_{s\in S_1} g(s) + \sum_{Z\in\mathcal{Z}} d(t_Z),\end{aligned}$$

where the last inequality follows from (3.6).

Therefore, to prove the theorem, it suffices to show that

$$\sum_{s\in S_1} g(s) + \sum_{Z\in\mathcal{Z}} d(t_Z) \le (3/2)\rho_{\mathrm{UFL}} \cdot opt(I). \tag{3.8}$$

For this, consider an arbitrary set $Z \in \mathcal{M}_2$. By the choice of terminal $t_Z \in Z$ and condition (ii) of Lemma 3.3, we have

$$\sum_{t\in Z} d(t) \ge |Z| d(t_Z) \ge (2/3)\kappa d(t_Z). \tag{3.9}$$

Now consider a pair of subsets $X_i, Y_i \in \mathcal{Z}$, and their terminals $t_Z = t_{X_i} \in X_i$ and $t_Z = t_{Y_i} \in Y_i$ chosen in Step 3. Assume without loss of generality $d(t_{X_i}) \leq d(t_{Y_i})$. Then t_{X_i} has the smallest distance among all terminals in $X_i \cup Y_i$. Hence for the set $Z_0 \subseteq X_i \cup Y_i$ of terminals with the first $(2/3)\kappa$ smallest vertex weight d, we have

$$\sum_{t \in Z_0} d(t) \geq (2/3)\kappa d(t_{X_i}).$$

For the set $(X_i \cup Y_i) - Z_0$ of the remaining terminals, we have

$$\begin{aligned} \sum_{t \in (X_i \cup Y_i) - Z_0} d(t) &\geq (|X_i| + |Y_i| - (2/3)\kappa) d(t_{Y_i}) \\ &\geq (2/3)\kappa d(t_{Y_i}), \end{aligned}$$

where the last inequality follows from $|X_i| + |Y_i| \geq (4/3)\kappa$ in condition (iii) of Lemma 3.3. Therefore, it holds

$$\sum_{t \in X_i \cup Y_i} d(t) \geq (2/3)\kappa d(t_{X_i}) + (2/3)\kappa d(t_{Y_i}). \tag{3.10}$$

By summing inequalities (3.9) and (3.10) overall subsets in $Z \in \mathcal{Z}$, we have

$$(2/3)\kappa \sum_{Z \in \mathcal{Z}} d(t_Z) \leq \sum_{t \in Z \in \mathcal{Z}} d(t) \leq \sum_{t \in M} d(t).$$

By (3.6), this implies

$$\begin{aligned} \sum_{s \in S_1} g(s) + \sum_{Z \in \mathcal{Z}} d(t_Z) &\leq \sum_{s \in S_1} g(s) + (3/2) \sum_{t \in M} (d(t)/\kappa) \\ &\leq (3/2)\rho_{\text{UFL}} \cdot opt(I), \end{aligned}$$

proving (3.8), as required. □

Chapter 4

The Minimum Cost Edge Installation Problem for Routings

In this chapter we study the problem of routing demands from a set of sources in an edge-weighted network to a single vertex designated as a sink such that the demand from each source is routed to the sink through a single path. Capacity can be installed on each edge by any amount which is multiples of a fixed quantity. The weight of the edge stands for the cost of installing one unit of the fixed quantity. The problem asks to install capacities on edges of the network to support the flow along paths at minimum cost.

4.1 Introduction

We study a problem of finding routings from a set of sources to a single sink in a network with an edge installing cost. This problem is a fundamental and economically significant one that arises in a hierarchical design of telecommunication networks [24] and transportation networks [65, 74]. In telecommunication networks this corresponds to installing transmission facilities such as fiber-optic cables, which represent the edges of the network. In other applications, optical cables may be replaced by pipes, trucks, and so on.

In this chapter, we study a special case of SSBB (defined in Section 1.6) that arises from transportation networks [74]. A multinational corporation wishes to enter a new geographic area, characterized by demand at each city. It has identified the location of its manufacturing facility. Suppose that the shipping of the goods will be carried out by some transport company. This transport company has only one truck type, with a fixed capacity. For each truck, the transport company charges at a fixed rate per mile, and offers no discount in the case where the truck is not utilized to full capacity. The problem facing the corporation is to decide a shipping plan of the finished goods to each city, so that the total demand at each city is met and the total cost is minimized.

In such a transportation network, we have a single cable type with a fixed capacity $\lambda > 0$

for all edges, and we are interested in constructing a set $\mathcal{P}$ of paths each of which connects one of given sources to a single sink s. The cost of installing a copy of an edge e is represented by the weight of e. A subset of paths in $\mathcal{P}$ can pass through a single copy of an edge e as long as the total demand of these paths does not exceed the edge capacity λ; any integer number of separated copies of e are allowed to be installed. However, the demand of each source is required to be sent to the sink s without being split at any vertex and without going through more than one copy of the same edge. The cost of a set $\mathcal{P}$ of paths is defined by the minimum cost of installing copies of edges such that the demand of each source can be routed to the sink under the edge capacity constraint; i.e., it is given by

$$cost(\mathcal{P}) = \sum_{e \in E(G)} h_{\mathcal{P}}(e) w(e).$$

where $h_{\mathcal{P}}(e)$ is the minimum number of copies of e required for routing the set of all demands along e, simultaneously. The goal is to find a set $\mathcal{P}$ of paths that minimizes $cost(\mathcal{P})$. We call this problem, the *minimum cost edge installation problem* (MCEI). Notice that, in order to get a feasible solution to MCEI, such edge capacity λ should be as much as the maximum demand in the network. MCEI can be formally defined as follows.

Minimum Cost Edge Installation Problem (MCEI):
Input: A connected graph $G = (V, E)$, an edge weight function $w : E \to R^+$, an edge capacity $\lambda > 0$, a sink $s \in V$, a set $M \subseteq V - \{s\}$ of sources, and a demand function $q : M \to R^+$ such that $q(v) \leq \lambda$, $v \in M$.
Feasible solution: A set $\mathcal{P} = \{P_v \mid v \in M, \{s, v\} \subseteq V(P_v)\}$ of paths in G.
Goal: Find a feasible solution $\mathcal{P}$ that minimizes $cost(\mathcal{P})$.

MCEI is closely related to the *capacitated network design problem* (CND), which can be stated as follows. We are given a connected graph G such that each edge $e \in E(G)$ is weighted by a nonnegative real $w(e)$, a vertex $s \in V(G)$ designated as a sink, and a subset $M \subseteq V(G) - \{s\}$ of sources. Each source $v \in M$ has a nonnegative demand $q(v)$, all of which must be routed to s through a single path. A cable with fixed capacity λ is available for installing on the edges of the graph, where installing i copies of the cable on edge e costs $iw(e)$ and provides $i\lambda$ capacity, which may be shared by demands of different sources. The capacity installed on an edge has to be at least as much as the total demand routed through this edge. CND asks to find a minimum cost installation of cables that provides a sufficient capacity to route all of the demand simultaneously to s.

For CND, Mansour and Peleg [53] gave an $O(\log n)$-approximation algorithm for a graph with n vertices. Salman et al. [65] designed a 7-approximation algorithm for CND based on a light approximate shortest path tree defined by Khuller et al. [44]. Afterwards Hassin et al. [33] gave a $(2 + \rho_{\mathrm{ST}})$-approximation algorithm. By using of a slight intricate version of this algorithm, they improved the approximation ratio to $(1 + \rho_{\mathrm{ST}})$ when every source has

unit demand. When all non-sink vertices are sources, the approximation ratio of Hassin et al. [33] becomes 3 for general demands and 2 for unit demands, since the Steiner tree problem in this case is a minimum spanning tree problem.

Note that, a solution to each of MCEI and CND can be characterized by specifying the path P_v for each source v along which the demand $q(v)$ of v will be sent to the sink. The cables installed on each edge of the network are induced by these paths. In particular, for each edge e, a feasible solution to MCEI assigns an integer number of separated cable copies required for routing all demands in $\{q(v) \mid e \in E(P_v)\}$, simultaneously. On the other hand, a feasible solution to CND assigns on e at least $\lceil \sum_{v:e\in E(P_v)} q(v)/\lambda \rceil$ copies of the cable. That is, on contrary to MCEI, CND allows the demand from a source to be split among different copies of the same edge. Note that, the algorithm of Hassin et al. [33] to CND takes the advantage (over MCEI) of this assumption only for routing demands larger than λ to the sink. Hence, their algorithms can be used to obtain approximate solutions to MCEI with approximation ratios $1 + \rho_{\mathrm{ST}}$ and $2 + \rho_{\mathrm{ST}}$ for the unit and general demand networks, respectively. In this chapter, we proved that there is a $(15/8 + \rho_{\mathrm{ST}})$-approximation algorithm to MCEI with general demands. Our result is based on a new and elaborated method for partitioning the source set of a given tree. When $M = V$, the approximation ratio of our algorithm becomes 2.875.

4.2 Preliminaries

This section introduces some definitions and lower bounds to MCEI. We first introduce a subgraph which plays a key role in our algorithm.

Definition 4.1. *For a vertex v in a rooted tree T, a source set $Z_v \subseteq V(T_v) - \{v\}$, a demand function $q : Z_v \to R^+$, and a positive number λ, a binary rooted tree T_v is said to be a* 4/7-balance-tree *if $q(Z_v) > \lambda$ holds and the total demand in each of the branches of T_v is less than $(4/7)\lambda$.*

The rest of this section presents two lower bounds on the optimal value to instances of MCEI. The first lower bound has been proved and used to derive approximation algorithms to CND [33].

Lemma 4.1. *For an instance $I = (G, w, \lambda, s, M, q)$ of MCEI, let $opt(I)$ be the weight of an optimal solution to I, and T^* be the minimum weight of a tree that spans $M \cup \{s\}$ in G. Then*

$$\max\left\{ w(T^*), \frac{1}{\lambda} \sum_{t\in M} q(t) d_{(G,w)}(s,t) \right\} \leq opt(I).$$

The second lower bound is derived from an observation on the distance from sources $t \in M$ with $q(t) > \lambda/2$ to sink s.

Lemma 4.2. *For an instance $I = (G, w, \lambda, s, M, q)$ of* MCEI*, let $opt(I)$ be the weight of an optimal solution to I, and define $M' = \{t \in M \mid q(t) > \lambda/2\}$. Then*

$$\sum_{t \in M'} d_{(G,w)}(s,t) \leq opt(I).$$

Proof. The inequality is immediate since for any two sources $u, v \in M'$, the paths P_u and P_v of an optimal solution cannot share the capacity of a single copy of any edge $e \in E$. □

Given an instance $I = (G, w, \lambda, s, M, q)$ of MCEI, our algorithm first produces a tree T of G that spans all vertices in $M \cup \{s\}$, finds a partition $\mathcal{M}$ of M, and assigns a vertex $t_Z \in Z$ for each subset $Z \in \mathcal{M}$ such that when all demands in each subset $Z \in \mathcal{M}$ are routed to t_Z simultaneously, the total flow on each edge of T is at most λ. We call such a vertex t_Z the *hub vertex* of Z. Afterward, for each $Z \in \mathcal{M}$, we install a copy for each edge in a shortest path $SP(s, t_Z)$ between s and t_Z in G, and extend the path between $t \in Z$ and t_Z in T to a path P_t from t to s by adding $SP(s, t_Z)$. The running time of this algorithm is dominated by the approximation algorithm for the Steiner tree problem to compute tree T.

4.3 Tree partition

The purpose of this section is to describe how to construct a "tree partition" in a tree that spans a source set. For a tree T with a source set $M \subseteq V(T)$, tree partition consists of finding a specific partition of M. Such a tree partition will be the basis of our approximation algorithm to MCEI in the next section. We first present some results for special cases of tree partitioning. In this section, we introduce a general nonnegative vertex weight function d on vertices v, where $d(v)$ will be defined in the main algorithm to be the distance between s and v.

4.3.1 Tree partition in special trees

In this subsection, we prepare several lemmas on tree partition problem for a tree with special structure.

For a technical reason, we consider the following instance $(T_x, \lambda, Z_x, q, d)$ in the next three lemmas. We are given a binary rooted tree T_x with an edge capacity $\lambda > 0$, a source set $Z_x = L(T_x)$, a demand function $q : Z_x \to R^+$ such that $q(t) \leq \lambda/2$ for all $t \in Z_x$, and a vertex weight function $d : Z_x \to R^+$. Moreover, for each $u \in Ch(x)$, we assume that

$$\text{either } q(V(T_u) \cap Z_x) < (4/7)\lambda, \text{ or}$$

$$T_u \text{ contains a 4/7-balance-tree and satisfies } q(V(T_u) \cap Z_x) < (8/7)\lambda. \tag{4.1}$$

We will show how to partition Z_x into subsets, and choose a hub vertex for each subset such that, when demands in each subset are routed to its hub vertex simultaneously, the total flow on each edge of T_x is bounded from above by λ.

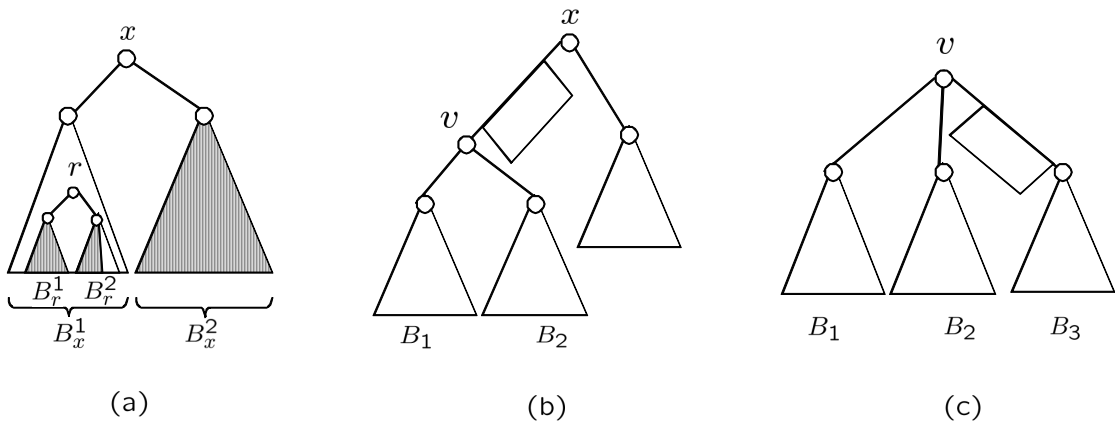

Figure 4.1: (a) illustration of Case 2 in Lemma 4.3, where L and N are represented by white and gray regions, respectively; (b) illustration of a tree T_x in Lemma 4.4; (c) a tree T' obtained by rerooting T_x at vertex v.

We start by the following basic lemma which will be used in proving the three lemmas in this section.

Lemma 4.3. *Given a tree $(T_x, \lambda, Z_x, q, d)$ such that each branch of T_x has more than $(3/7)\lambda$ and less than $(4/7)\lambda$ demand, there is a partition $\{L, N\}$ of Z_x such that $q(N) \geq (5/7)\lambda$, and when the demands in L and N are routed to $t_L = x$ and $t_N = \operatorname{argmin}\{d(t) \mid t \in N\}$, respectively, the total amount of these flow on each edge of T_x is at most λ.*

Proof. Roughly speaking, we choose a set of sources of at least $(5/7)\lambda$ demand, route its demand to a source of the minimum weight d in the set, and then route the remaining demand to the root x. In particular, such a partition is constructed based on the structure of the tree in order to guarantee that the edge capacity λ remains satisfied.

Let B_x^1 and B_x^2 denote the two branches of T_x, and let $Z_x^i = V(B_x^i) \cap Z_x$, $i = 1, 2$. Let t_1 and t_2 denote the sources of the largest demands in Z_x^1 and Z_x^2, respectively. We have the following two cases.

Case 1. $q(t_1) + q(t_2) \geq (5/7)\lambda$: Let $N = \{t_1, t_2\}$ and $L = Z_x - N$. We have $q(N) = q(t_1) + q(t_2) \leq \lambda/2 + \lambda/2 = \lambda$ and $q(L) = q(Z_x) - q(N) < (8/7)\lambda - (5/7)\lambda = (3/7)\lambda$. Now, let the demands in L and N be routed to t_L and t_N, respectively. The flow on each edge in $E(T_x)$ corresponding to the sources assigned to L is at most $q(L) < (3/7)\lambda$, and the flow on each edge in $E(T_x)$ corresponding to the sources assigned to N is at most $\lambda/2$ (since $q(t_1), q(t_2) \leq \lambda/2$). Hence, the total flow on each edge of T_x is bounded from above by λ.

Case 2. $q(t_1) + q(t_2) < (5/7)\lambda$: Then $q(t_1) < (5/14)\lambda$ or $q(t_2) < (5/14)\lambda$, where $q(t_1) < (5/14)\lambda$ is assumed without loss of generality. Find a vertex $r \in V(B_x^1)$ with the maximum depth in B_x^1 such that $q(V(T_r) \cap Z_x) \geq (2/7)\lambda$ (possibly $r \in L(T_x)$). Such a vertex r is

well defined since $q(Z_1) > (3/7)\lambda$. Let B_r^1 and B_r^2 denote the two branches of T_r and let $Z_r^i = V(B_r^i) \cap Z_x$, $i = 1, 2$, such that Z_r^1 contains the source of the smallest weight d in T_r. Choose a minimal set N such that $Z_x^2 \cup Z_r^1 \subseteq N \subseteq Z_x^2 \cup Z_r^1 \cup Z_r^2$ and $q(N) \geq (5/7)\lambda$ (see Fig. 4.1(a)). Such a set N is well defined since $q(Z_x^2) + q(V(T_r) \cap Z_x) > (3/7)\lambda + (2/7)\lambda = (5/7)\lambda$. Note that $q(N) < \lambda$ since $q(Z_x^2) < (4/7)\lambda$ and $q(Z_r^1), q(Z_r^2) < (2/7)\lambda$ (by the choice of r). Let $L = Z_x - N$. We have $q(L) = q(Z_x) - q(N) < (8/7)\lambda - (5/7)\lambda = (3/7)\lambda$. Now, let the demands in L and N be routed to t_L and t_N, respectively. By the choice of Z_r^1, it holds $t_N \in Z_r^1 \cup Z_x^2$.

First, assume that $t_N \in Z_r^1$. The flow on each edge in $E(B_r^2)$ is at most $q(Z_r^2) < (2/7)\lambda$, and the flow on each edge in $E(B_r^1)$ (all corresponding to the sources assigned to N) is at most $q(N) < \lambda$. Also, the flow on each edge in $E(B_x^1) - E(T_r)$ corresponding to the sources assigned to N (resp., L) is at most $q(Z_x^2) < (4/7)\lambda$ (resp., $q(L) < (3/7)\lambda$). Finally, we see that the flow on each edge in $E(B_x^2)$ (all corresponding to the sources assigned to N) is at most $q(Z_x^2) < (4/7)\lambda$.

Next, assume that $t_N \in Z_x^2$. We observe that the total flow on each edge in $E(B_x^1)$ is at most $q(Z_x^1) < (4/7)\lambda$, and the flow on each edge in $E(B_x^2)$ (all corresponding to the sources assigned to N) is at most $q(N) < \lambda$. Hence, the total flow on each edge of T_x is bounded from above by λ. □

For a tree T_x with $(8/7)\lambda \leq q(Z_x) < (12/7)\lambda$, the following lemma partitions Z_x into two subsets such that either (i) the demand of each of the two subsets is at least $(4/7)\lambda$, or (ii) the demand of one subset is at least $(4/7)\lambda$ and the other subset contains a source of the minimum weight d in T_x.

Lemma 4.4. *Given a tree $(T_x, \lambda, Z_x, q, d)$ with $(8/7)\lambda \leq q(Z_x) < (12/7)\lambda$, there is a partition $\{X, Y\}$ of Z_x such that one of the following holds:*

(i) *There is a subset $Y' \subseteq Y$ such that $\min\{q(Y'), q(X)\} \geq (4/7)\lambda$, and when the demands in X and Y are routed to $t_X = \operatorname{argmin}\{d(t) \mid t \in X\}$ and $t_Y = \operatorname{argmin}\{d(t) \mid t \in Y'\}$ along T_x, respectively, the total amount of these flow on each edge of T_x is at most λ.*

(ii) *$q(Y) \geq (4/7)\lambda$, and when the demands in X and Y are routed to $t_X = \operatorname{argmin}\{d(t) \mid t \in Z_x\}$ and $t_Y = \operatorname{argmin}\{d(t) \mid t \in Y\}$ along T_x, respectively, the total amount of these flow on each edge of T_x is at most λ.*

Proof. By the assumptions on T_x, the inequality $(8/7)\lambda \leq q(Z_x) < (12/7)\lambda$ implies that one branch of T_x contains a 4/7-balance-tree, say T_v, and has less than $(8/7)\lambda$ demand, and the other branch has less than $(4/7)\lambda$ demand. Regard T_x as a tree T' rooted at v, and let B_1, B_2, and B_3 denote the three branches of T', where $T_v = B_1 + B_2$. Let $Z_i = V(B_i) \cap Z_x$, $i = 1, 2, 3$ (see Fig. 4.1(b)-(c)). Note that $q(Z_3) = q(Z_x) - q(Z_1 \cup Z_2) < (12/7)\lambda - \lambda = (5/7)\lambda$ and $\min\{q(Z_1), q(Z_2)\} > (3/7)\lambda$, by the definition of the 4/7-balance-tree. Now we distinguish

the following cases.

Case 1. $q(Z_3) \geq (4/7)\lambda$: We show that (i) holds in this case. We apply Lemma 4.3 to $B_1 + B_2$ to obtain a partition $\{L, N\}$ of $Z_1 \cup Z_2$ that satisfies the lemma. Let $X = N$ and $Y = Z_x - X$, and choose $Y' = Z_3$. Now, let the demands in X and Y be routed to t_X and t_Y, respectively. By Lemma 4.3, the flow on each edge in $E(B_1) \cup E(B_2)$ is at most λ. Moreover, the flow on each edge in $E(B_3)$ is at most $q(Y) = q(Z_x) - q(X) < (12/7)\lambda - (5/7)\lambda = \lambda$.

Case 2. $q(Z_3) < (4/7)\lambda$: We show that (ii) holds in this case. Assume without loss of generality that $t_X \in Z_1$. If $q(Z_2 \cup Z_3) \leq \lambda$, then $X = Z_1$ and $Y = Z_x - X$ satisfy that $q(X) = q(Z_1) < (4/7)\lambda$, $q(Y) = q(Z_x) - q(X) > (8/7)\lambda - (4/7)\lambda = (4/7)\lambda$, and $E(T\langle X\rangle) \cap E(T\langle Y\rangle) = \emptyset$, implying that when the demands in X and Y are routed to t_X and t_Y, respectively, the flow on each edge of T' is at most λ. Assume that $q(Z_2 \cup Z_3) > \lambda$. This implies that $q(Z_3) > \lambda - q(Z_2) > \lambda - (4/7)\lambda = (3/7)\lambda$. We apply Lemma 4.3 to $B_2 + B_3$ to obtain a partition $\{L, N\}$ of $Z_2 \cup Z_3$ that satisfies the lemma. Then $Y = N$ and $X = Z_x - Y$ satisfy $(5/7)\lambda \leq q(Y) \leq \lambda$ and $q(X) = q(Z_x) - q(Y) < (12/7)\lambda - (5/7)\lambda = \lambda$. Now, let the demands in X and Y be routed to t_X and t_Y, respectively. By Lemma 4.3, the flow on each edge in $E(B_2) \cup E(B_3)$ is at most λ. Moreover, the flow on each edge in $E(B_1)$ (all corresponding to the sources assigned to X) is at most $q(X) < \lambda$. □

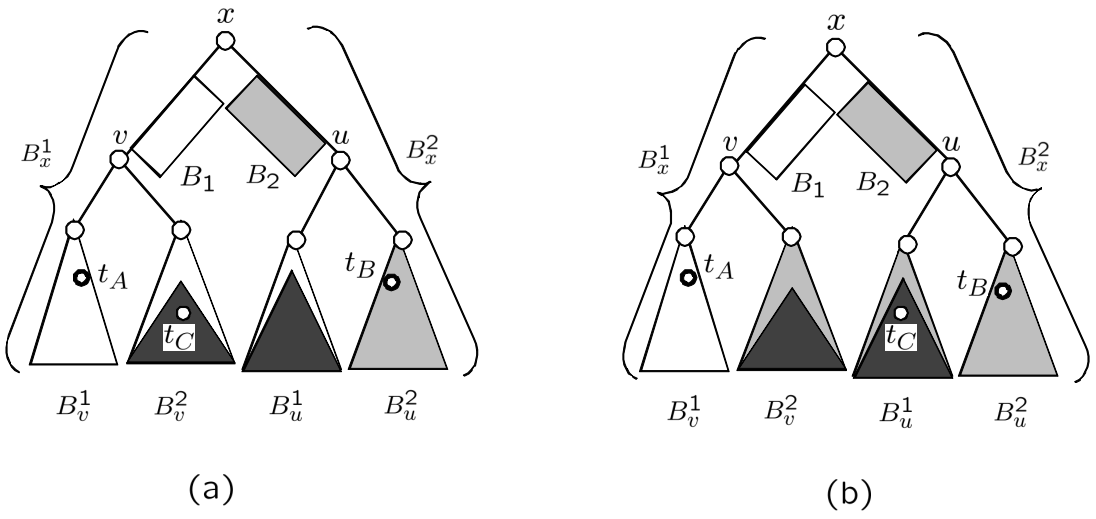

Figure 4.2: Illustration of Case 2 in Lemma 4.5, where A, B, and C are represented by white, gray, and black regions, respectively; (a) $t_C \in Z_v^2$; (b) $t_C \in Z_u^1$.

For a tree T_x with $q(Z_x) \geq (12/7)\lambda$, the following lemma partitions the vertex set Z_x of a tree T_x into three subsets so that, the first subset contains a source of the minimum weight d in Z_x, the second subset contains a source of the minimum weight d among the remaining sources and has more than $(3/7)\lambda$ demand, and the last subset has at least $(5/7)\lambda$ demand.

Lemma 4.5. *Given a tree $(T_x, \lambda, Z_x, q, d)$ with $q(Z_x) \geq (12/7)\lambda$, there is a partition $\{A, B, C\}$ of Z_x such that $q(B) > (3/7)\lambda$, $q(C) \geq (5/7)\lambda$, and when the demands in A, B, and C are routed to $t_A = \mathrm{argmin}\{d(t) \mid t \in Z_x\}$, $t_B = \mathrm{argmin}\{d(t) \mid t \in Z_x - A\}$, and $t_C = \mathrm{argmin}\{d(t) \mid t \in C\}$, respectively, the total amount of these flow on each edge of T_x is at most λ.*

Proof. We first describe the structure of T_x. Let B_x^1 and B_x^2 denote the two branches of T_x, and let $Z_x^i = V(B_x^i) \cap Z_x$, $i = 1, 2$. Then by $q(Z_x) \geq (12/7)\lambda$ and the assumptions on T_x, B_x^i contains a 4/7-balance-tree and $q(Z_x^i) < (8/7)\lambda$ holds for each $i = 1, 2$. Let T_v and T_u denote the 4/7-balance-trees of B_x^1 and B_x^2, respectively. Let $Z_v = V(T_v) \cap Z_x$ and $Z_u = V(T_u) \cap Z_x$, and denote the two branches of T_v (resp., T_u) by B_v^1 and B_v^2 (resp., B_u^1 and B_u^2). Let $Z_v^i = V(B_v^i) \cap Z_x$ and $Z_u^i = V(B_u^i) \cap Z_x$, $i = 1, 2$. Let B_1 (resp., B_2) denote the tree obtained from B_x^1 (resp., B_x^2) by deleting vertices in $D(v) - \{v\}$ (resp., $D(u) - \{u\}$), and let $Z_i = V(B_i) \cap Z_x$, $i = 1, 2$. See Fig. 4.2. Note that $q(Z_1) = q(Z_x^1) - q(Z_v) < (8/7)\lambda - \lambda = (1/7)\lambda$. Similarly $q(Z_2) < (1/7)\lambda$. Also, $\min\{q(Z_v^i), q(Z_u^i)\} > (3/7)\lambda$, $i = 1, 2$, by the definition of the 4/7-balance-tree. We distinguish the following cases.

Case 1. $t_A \in Z_1 \cup Z_2$: We apply Lemma 4.3 to T_v (resp., T_u) to obtain a partition $\{L_v, N_v\}$ (resp., $\{L_u, N_u\}$) of Z_v (resp., Z_u) that satisfies the lemma. Let $B = N_v$, $C = N_u$, and $A = Z_1 \cup Z_2 \cup L_v \cup L_u$, where $t_B \in N_v$ is assumed without loss of generality. Note that $q(Z_1 \cup L_v) = q(Z_x^1) - q(N_v) < (8/7)\lambda - (5/7)\lambda = (3/7)\lambda$. Similarly, $q(Z_2 \cup L_u) < (3/7)\lambda$. Therefore, $q(A) < (6/7)\lambda$. Now, let the demands in A, B, and C be routed to t_A, t_B, and t_C, respectively. By Lemma 4.3, the flow on each edge in $E(T_v) \cup E(T_u)$ is at most λ. Also, the flow on each edge in $E(B_1) \cup E(B_2)$ (all corresponding to the sources assigned to A) is at most $q(A) < (6/7)\lambda$.

Case 2. $t_A \in Z_v^1$, and Z_u^2 contains the source of the minimum weight d in $Z_x - (Z_v^1 \cup Z_1)$: We apply Lemma 4.3 to the minimal subtree of T_x spanning $Z_v^2 \cup Z_u^1$ to obtain a partition $\{L, N\}$ of $Z_v^2 \cup Z_u^1$ that satisfies the lemma. Let $C = N$. If $t_C \in Z_u^1$, then $A = Z_v^1 \cup Z_1$ and $B = Z_u^2 \cup Z_2 \cup L$ satisfy $q(A) = q(Z_v^1) + q(Z_1) < (4/7)\lambda + (1/7)\lambda = (5/7)\lambda$ and $q(B) = q(Z_x^2) + q(Z_v^2) - q(N) < (8/7)\lambda + (4/7)\lambda - (5/7)\lambda = \lambda$. Otherwise ($t_C \in Z_v^2$), $A = Z_v^1 \cup Z_1 \cup L$ and $B = Z_u^2 \cup Z_2$ satisfy $q(A) = q(Z_x^1) + q(Z_u^1) - q(N) < \lambda$ and $q(B) = q(Z_u^2) + q(Z_2) < (5/7)\lambda$. In both cases, $q(B) \geq q(Z_u^2) > (3/7)\lambda$ (since $Z_u^2 \subseteq B$). Now, let the demands in A, B, and C be routed to t_A, t_B, and t_C, respectively. By Lemma 4.3, the flow on each edge in $E(B_v^2) \cup E(B_u^1)$ is at most λ. Also, the flow on each edge in $E(B_v^1)$ (all corresponding to the sources assigned to A) is at most $q(A) < \lambda$, and the flow on each edge in $E(B_u^2)$ (all corresponding to the sources assigned to B) is at most $q(B) < \lambda$. The flow on each edge in $E(B_1) \cup E(B_2)$ is less than $(6/7)\lambda$ since $\max\{q(Z_1), q(Z_2)\} < (1/7)\lambda$ and $\max\{q(Z_v^2), q(Z_u^1)\} < (4/7)\lambda$. Hence, the total flow on each edge of T_x is bounded from above by λ.

Case 3. $t_A \in Z_v^1$, and Z_2 contains the source of the minimum weight d in $Z_x - (Z_v^1 \cup Z_1)$: Apply Lemma 4.3 to T_u to obtain a partition $\{L, N\}$ of Z_u that satisfies the lemma. Then let $C = N$, $A = Z_v^1 \cup Z_1$, and $B = Z_v^2 \cup Z_2 \cup L$ (see Fig. 4.3). Note that $q(L \cup Z_2) = q(Z_x^2) - q(N) < (8/7)\lambda - (5/7)\lambda = (3/7)\lambda$. Therefore, $q(A) = q(Z_v^1) + q(Z_1) < (4/7)\lambda + (1/7)\lambda = (5/7)\lambda$ and $(3/7)\lambda < q(B) = q(Z_v^2) + q(Z_2 \cup L) < (4/7)\lambda + (3/7)\lambda = \lambda$. Now, let $q(A)$, $q(B)$, and $q(C)$ be routed to t_A, t_B, and t_C, respectively. By Lemma 4.3, the flow on each edge in $E(T_u)$ is at most λ. Also, the flow on each edge in $E(B_v^1)$ (all corresponding to the sources assigned to A) is at most $q(A) < (5/7)\lambda$, and the flow on each edge in $E(B_u^2)$ is at most $q(B_v^2) < (4/7)\lambda$. Finally, we observe that the total flow on each edge in $E(B_1)$ is at most $q(Z_v^2) + q(Z_1) < (4/7)\lambda + (1/7)\lambda = (5/7)\lambda$, and the flow on each edge in $E(B_2)$ (all corresponding to the sources assigned to B) is at most $q(B) < \lambda$. Hence, the total flow on each edge of T_x is bounded from above by λ. □

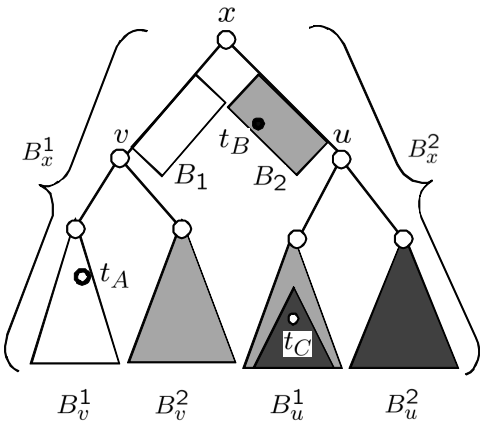

Figure 4.3: Illustration of Case 3 in Lemma 4.5, where A, B, and C are represented by white, gray, and black regions, respectively.

Lemma 4.6. *Given a tree $(T_x, \lambda, Z_x, q, d)$ with $Z_x \neq \emptyset$, there is a family $\mathcal{Z}$ of at most two disjoint subsets of Z_x such that $q(Z) \geq (5/7)\lambda$ for each $Z \in \mathcal{Z}$ and when the demands in each subset $Z \in \mathcal{Z}$ is routed to $t_Z = \operatorname{argmin}\{d(t) \mid t \in Z\}$ and all demands in $Z_x - \cup_{Z \in \mathcal{Z}} Z$ are routed to x, the total amount of these flow on each edge of T_x is at most λ.*

Proof. For each branch B of T_x with less than $(4/7)\lambda$ demand, let the demands in B be routed to x. Obviously, the flow on each edge in $E(B)$ is less than $(4/7)\lambda$. For each branch B of T_x that contains a 4/7-balance-tree and has less than $(8/7)\lambda$ demand, we proceed as follows. Let $Z = V(B) \cap Z_x$. Denote the 4/7-balance-tree of B by T_v and its source set by Z_v. We apply Lemma 4.3 to T_v to obtain a partition $\{L, N\}$ of Z_v that satisfies the lemma. Note that $q(Z - N) = q(Z) - q(N) < (8/7)\lambda - (5/7)\lambda = (3/7)\lambda$. Add N to $\mathcal{Z}$. Now, let the demands in N and $Z - N$ be routed to t_N and x, respectively. By Lemma 4.3, the flow

on each edge in $E(T_v)$ is at most λ. Also, the the flow on each edge in $E(B) - E(T_v)$ (all corresponding to the sources assigned to $Z - N$) is at most $q(Z - N) < (3/7)\lambda$. □

4.3.2 Algorithm for tree partition

In this subsection, we present an algorithm that exploits the results in Lemmas 4.4-4.6 to compute a partition of the source set of a general tree given in the next theorem.

Lemma 4.7. *Given a tree T rooted at s, an edge capacity $\lambda > 0$, a source set $M \subseteq V(T)$, a demand function $q : M \to R^+$ such that $q(t) \leq \lambda/2$, $t \in M$, and a vertex weight function $d : M \to R^+$, there is a partition $\mathcal{M} = \{Z_0\} \cup \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$ of M, where $\mathcal{M}_2 = \cup_{1 \leq i \leq k}\{X_i, Y_i\}$ and $\mathcal{M}_3 = \cup_{1 \leq i \leq \ell}\{A_i, B_i, C_i\}$, and a set $\mathcal{H} = \{t_Z \in Z \mid Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3\} \cup \{t_{Z_0} = s\}$ of hub vertices, that satisfy:*

(i) *For each subset $Z \in \mathcal{M}_1$, there is a subset $Z' \subseteq Z$ with $q(Z') \geq (4/7)\lambda$, and $t_Z = \mathrm{argmin}\{d(t) \mid t \in Z'\}$.*

(ii) *For $i = 1, 2, \ldots, k$, $q(Y_i) \geq (4/7)\lambda$, $q(X_i \cup Y_i) \geq (8/7)\lambda$, $t_{X_i} = \mathrm{argmin}\{d(t) \mid t \in X_i \cup Y_i\}$, and $t_{Y_i} = \mathrm{argmin}\{d(t) \mid t \in Y_i\}$.*

(iii) *For $i = 1, 2, \ldots, \ell$, $q(B_i) > (3/7)\lambda$, $q(C_i) \geq (5/7)\lambda$, $q(A_i \cup B_i \cup C_i) \geq (12/7)\lambda$, and $t_{A_i} = \mathrm{argmin}\{d(t) \mid t \in Z_x\}$, $t_{B_i} = \mathrm{argmin}\{d(t) \mid t \in Z_x - A_i\}$, and $t_{C_i} = \mathrm{argmin}\{d(t) \mid t \in C_i\}$.*

(iv) *When the total demand of each subset $Z \in \mathcal{M}$ is routed to t_Z simultaneously, the total amount of these flow on each edge of T is bounded from above by λ.*

Furthermore, such a partition $\mathcal{M}$ can be computed in polynomial time. □

To prove Lemma 4.7, we can assume without loss of generality that in a given tree T, (i) all sources are leaves, i.e., $M = L(T)$, by introducing a new edge of weight zero for each non-leaf source, and (ii) $|Ch(v)| = 2$ holds for every non-leaf $v \in V(T)$, i.e., T is a binary tree rooted at s, by splitting vertices of degree more than 3 with new edges of zero weights.

We prove Lemma 4.7 by showing that the next algorithm actually delivers a desired partition $\mathcal{M} = \{Z_0\} \cup \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$. We first choose a vertex $v \notin Q \cup \{s\}$ with the maximum depth in the current tree such that the total demand of a source set Z_v of the tree rooted at v is at least $(4/7)\lambda$, where Q is initialized to be empty and is used to keep track of vertices v in the current tree such that T_v contains a 4/7-balance-tree and satisfies $q(Z_v) < (8/7)\lambda$. Depending on the total demand of Z_v, we add Z_v to $\mathcal{M}_1$, add the vertex v to Q, or compute a partition of Z_v by using Lemma 4.4 or 4.5. In the latter case, we add the subsets of the obtained partition to one of $\mathcal{M}_2$ and $\mathcal{M}_3$. We then remove all sources in $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$ from M and repeat these steps on the minimal subtree of T that spans s and the current

source set until there is no such vertex v. Finally, by using Lemma 4.6 to the current tree, we construct at most two disjoint subsets of the current M, add them to $\mathcal{M}_1$, and let the remaining sources form Z_0. A formal description of the algorithm is given as follows.

Algorithm TreePartition
Input: A binary tree $\widehat{T}$ rooted at s, a capacity λ of each edge, a set $M = L(\widehat{T})$ of sources, a demand function $q : M \to R^+$ such that $q(t) \leq \lambda/2$, $t \in M$, and a vertex weight function $d : M \to R^+$.
Output: A pair $(\mathcal{M}, \mathcal{H})$ that satisfies the conditions in Lemma 4.7.
Initialize $T := \widehat{T}$; $Q := \mathcal{H} := \mathcal{M}_1 := \mathcal{M}_2 := \mathcal{M}_3 := \emptyset$.
1. **while** there exists a vertex $v \in V(T) - \{s\} - Q$ such that
   $q(V(T_v) \cap M) \geq (4/7)\lambda$ **do**
2. Choose such v with the maximum depth from s;
3. Let $Z_v := D_T(v) \cap M$; $T_v := T\langle Z_v \rangle$;
4. **begin** /* Distinguish the next four cases. */
5. **Case-1** $q(Z_v) \leq \lambda$: Let $\mathcal{M}_1 := \mathcal{M}_1 \cup \{Z_v\}$;
6. $t_{Z_v} = \text{argmin}\{d(t) \mid t \in Z_v\}$; $\mathcal{H} := \mathcal{H} \cup \{t_{Z_v}\}$;
7. **Case-2** $\lambda < q(Z_v) < (8/7)\lambda$: Let $Q := Q \cup \{v\}$;
8. **Case-3** $(8/7)\lambda \leq q(Z_v) < (12/7)\lambda$:
9. Apply Lemma 4.4 to $(T_v, \lambda, Z_v, q, d)$ to get a partition $\{X, Y\}$ of Z_v and vertices t_X and t_Y that satisfy the conditions in the lemma;
10. If Lemma 4.4(i) holds, then $\mathcal{M}_1 := \mathcal{M}_1 \cup \{X, Y\}$; $\mathcal{H} := \mathcal{H} \cup \{t_X, t_Y\}$;
11. If Lemma 4.4(ii) holds, then $\mathcal{M}_2 := \mathcal{M}_2 \cup \{X, Y\}$; $\mathcal{H} := \mathcal{H} \cup \{t_X, t_Y\}$;
12. **Case-4** $(12/7)\lambda \leq q(Z_v) < (16/7)\lambda$:
13. Apply Lemma 4.5 to $(T_v, \lambda, Z_v, q, d)$ to get a partition $\{A, B, C\}$ of Z_v and vertices t_A, t_B, and t_C that satisfy the conditions in the lemma;
14. $\mathcal{M}_3 := \mathcal{M}_3 \cup \{A, B, C\}$; $\mathcal{H} := \mathcal{H} \cup \{t_A, t_B, t_C\}$
15. **end**; /* Cases-1,2,3,4 */
16. Let $M := M - (\mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3)$; $T := T\langle M \cup \{s\} \rangle$
17. **endwhile**;
18. **if** $M = \emptyset$ **then**
19. $\{Z_0\} := \emptyset$
20. **else** /* $M \neq \emptyset$ */
21. Regard T as a tree T_s rooted at s and apply Lemma 4.6 to (T_s, λ, M, q, d) to get a family $\mathcal{Z}$ of at most two disjoint subsets of M and a vertex t_Z for each $Z \in \mathcal{Z}$ that satisfy the conditions in the lemma;
22. $Z_0 := M - \cup_{Z \in \mathcal{Z}} Z$; $t_{Z_0} := s$; $\mathcal{M}_1 := \mathcal{M}_1 \cup \mathcal{Z}$; $\mathcal{H} := \mathcal{H} \cup \{t_Z \mid Z \in \mathcal{Z} \cup \{Z_0\}\}$
23. **endif**.

Proof of Lemma 4.7. We first prove by induction the correctness of algorithm TreePartition. We first consider the vertex v chosen in the first iteration of the while-loop. By the choice of v, $q(V(T_u) \cap M) < (4/7)\lambda$ for all $u \in Ch(v)$. Hence $(4/7)\lambda \leq q(Z_v) < (8/7)\lambda$ holds, which implies that $q(Z_v) \leq \lambda$ or $\lambda < q(Z_v) < (8/7)\lambda$ can occur in the first iteration. That is, T_v satisfies the condition in (4.1) for a vertex v chosen in the first iteration. If $q(Z_v) \leq \lambda$ holds, then Z_v is removed from M and added to $\mathcal{M}_1$. Otherwise $\lambda < q(Z_v) < (8/7)\lambda$ holds and hence T_v is a 4/7-balance-tree. In the latter case, v is added to a set Q.

Assume that the algorithm works correctly after the execution of the jth iteration, and let T be the current tree. We show the correctness of the algorithm during the execution of the $(j+1)$st iteration. Note that, for any vertex v chosen by the algorithm, Z_v will be removed from the current M except for the case where $\lambda < q(Z_v) < (8/7)\lambda$. Now let v be a vertex selected in the $(j+1)$st iteration. Then we see that, for each child $u \in Ch(v)$, either (i) $q(V(T_u) \cap M) < (4/7)\lambda$ holds (if u has not been chosen before by the algorithm) or (ii) $u \in Q$ holds and T_u contains a 4/7-balance-tree and satisfies $q(V(T_u) \cap M) < (8/7)\lambda$ (otherwise). That is, T_v satisfies the condition in (4.1) for a vertex v chosen in the $(j+1)$st iteration. Therefore, one of $(4/7)\lambda \leq q(Z_v) \leq \lambda$, $\lambda < q(Z_v) < (8/7)\lambda$, $(8/7)\lambda \leq q(Z_v) < (12/7)\lambda$, and $(12/7)\lambda \leq q(Z_v) < (16/7)\lambda$ holds. Let B_v^1 and B_v^2 denote the two branches of T_v, and let Z_v^i denote the set of sources in B_v^i, $i = 1, 2$, where $q(Z_v^1) \geq q(Z_v^2)$. Now if $q(Z_v) \leq \lambda$ holds, then Z_v is removed from the current M after it is added to $\mathcal{M}_1$. If $\lambda < q(Z_v) < (8/7)\lambda$ holds, then T_v is a 4/7-balance-tree (if $q(Z_v^1), q(Z_v^2) < (4/7)\lambda$) or B_v^1 (consequently T_v) contains a 4/7-balance-tree (by $q(Z_v^1) \geq q(Z_v^2)$). In this case, the vertex v is added to a set Q. Finally, if $(8/7)\lambda \leq q(Z_v) < (12/7)\lambda$ (resp., $(12/7)\lambda \leq q(Z_v) < (16/7)\lambda$) holds then T_v satisfies conditions of Lemma 4.4 (resp., Lemma 4.5) in this case. In the latter two cases, Z_v is removed from the current M after elements of its partition are added to appropriate subsets of $\mathcal{M}$. Therefore, the algorithm works correctly during the execution of all iterations of the while-loop.

After the final iteration, there is no vertex $v \in V(T) - \{s\} - Q$ such that $q(V(T_v) \cap M) \geq (4/7)\lambda$ for the current tree T. If the current $M \neq \emptyset$, then for each child $u \in Ch(s)$, either (i) $q(V(T_u) \cap M) < (4/7)\lambda$ holds (if u has not been chosen before by the algorithm) or (ii) $u \in Q$ holds and T_u contains a 4/7-balance-tree and satisfies $q(V(T_u) \cap M) < (8/7)\lambda$ (otherwise). That is, the current tree T satisfies the condition in (4.1) and a desired partition of the current M can be constructed by using Lemma 4.6.

Now we prove that the partition obtained from algorithm TreePartition satisfies conditions (i)-(iv) in Lemma 4.7. Conditions (i)-(iii) follow immediately from construction of $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$. Now we show (iv). Let v be the vertex chosen in line 2 of an arbitrary iteration of the algorithm, where the subtree T_v of the current tree T is being processed in this iteration. Now, if Case-2 holds, then the algorithm just adds v to Q and then moves to the next iteration (the current M and T remain unchanged in this iteration). Otherwise (Case-1, 3, or 4 holds), the algorithm partitions the set Z_v of all sources of T_v into subsets and

chooses a hub vertex from each of these subsets. We then remove Z_v from the current source set M, that is, none of the vertices of T_v will become a hub vertex in the subsequent iterations of the algorithm. Thus it is sufficient to show that, overall iterations of the algorithm, when the demand of each source in Z_v is routed to its hub vertex simultaneously, the total flow on each edge of T_v is bounded from above by λ. Hence (iv) follows from the conditions of Lemmas 4.4, 4.5, and 4.6. This completes the correctness of TreePartition and the proof of Lemma 4.7. □

4.4 Approximation algorithm to MCEI

This section describes a framework of our approximation algorithm for MCEI and then analyzes its approximation ratio. The algorithm relies on the results on tree partition provided in Section 4.3.

Algorithm ApproxMCEI
Input: An instance $I = (G, w, \lambda, s, M, q)$ of MCEI.
Output: A solution $\mathcal{P}$ to I.

Step 1. Compute a Steiner tree T that spans $M \cup \{s\}$ in (G, w).
Regard T as a tree rooted at s, and define $d : M \to R^+$ by setting

$$d(t) := d_{(G,w)}(s, t), \quad t \in M.$$

Step 2. Let $M' := \{t \in M \mid q(t) > \lambda/2\}$.
For each $t \in M'$, choose a shortest path $SP(s, t)$ between s and t in (G, w), join t to s by installing a copy of each edge in $SP(s, t)$, and let $P_t := SP(s, t)$.

Step 3. Apply Lemma 4.7 to $(T, \lambda, s, M - M', q, d)$ to obtain a partition

$$\mathcal{M} = \{Z_0\} \cup \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$$

of $M - M'$, where $\mathcal{M}_2 = \cup_{1 \le i \le k}\{X_i, Y_i\}$ and $\mathcal{M}_3 = \cup_{1 \le i \le \ell}\{A_i, B_i, C_i\}$, and a set $\mathcal{H} = \{t_Z \in Z \mid Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3\} \cup \{t_{Z_0} = s\}$ of hub vertices, that satisfy conditions (i)-(iv) of the theorem.

Step 4. For each $t \in Z_0$, let P_t be the path between t and s in T.
For each $Z \in \mathcal{M} - \{Z_0\}$,

> Choose a shortest path $SP(s, t_Z)$ between s and t_Z in (G, w) and join t_Z to s by installing a copy of each edge in $SP(s, t_Z)$.
> For each $t \in Z$, let P_t be the path obtained from the path between t and t_Z in T by adding $SP(s, t_Z)$.

Step 5. Output $\mathcal{P} := \{P_t \mid t \in M\}$. □

Before analyzing the worst case performance of this algorithm, we show the following lemma. We use the conditions of a partition $\mathcal{M} = \{Z_0\} \cup \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$ and the set $\mathcal{H} = \{t_Z \in Z \mid Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3\} \cup \{t_{Z_0} = s\}$ of associated hub vertices defined in Lemma 4.7 to find an upper bound on $\sum_{Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z)$. This defines an upper bound on the total weight of edges installed in Step 4 of the algorithm.

Lemma 4.8. *Let $\mathcal{M} = \{Z_0\} \cup \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$ be a partition obtained by applying Lemma 4.7 to (T, λ, s, M, q, d), and let $\mathcal{H} = \{t_Z \in Z \mid Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3\} \cup \{t_{Z_0} = s\}$ be the associated set of hub vertices. Then it holds*

$$\sum_{t \in Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} q(t)d(t) \geq (4/7)\lambda \sum_{Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z).$$

Proof. First, consider a subset $Z \in \mathcal{M}_1$. Condition (i) of Lemma 4.7 implies that there is a subset $Z' \subseteq Z$ such that

$$\sum_{t \in Z} q(t)d(t) \geq \sum_{t \in Z'} q(t)d(t) \geq q(Z')d(t_Z) \geq (4/7)\lambda d(t_Z). \tag{4.2}$$

Now, consider a pair of subsets $X_i, Y_i \in \mathcal{M}_2$ defined in Lemma 4.7(ii). We have

$$\begin{aligned} \sum_{t \in X_i} q(t)d(t) + \sum_{t \in Y_i} q(t)d(t) &\geq q(X_i)d(t_{X_i}) + q(Y_i)d(t_{Y_i}) \\ &\geq (4/7)\lambda(d(t_{X_i}) + d(t_{Y_i})), \end{aligned} \tag{4.3}$$

since $q(Y_i) \geq (4/7)\lambda$, $q(X_i \cup Y_i) \geq (8/7)\lambda$, and $d(t_{X_i}) \leq d(t_{Y_i})$ hold.

Finally, consider a triple of subsets $A_i, B_i, C_i \in \mathcal{M}_3$ defined in Lemma 4.7(iii). We have

$$\begin{aligned} &\sum_{t \in A_i} q(t)d(t) + \sum_{t \in B_i} q(t)d(t) + \sum_{t \in C_i} q(t)d(t) \\ &\qquad \geq q(A_i)d(t_{A_i}) + q(B_i)d(t_{B_i}) + q(C_i)d(t_{C_i}) \\ &\qquad \geq (4/7)\lambda(d(t_{A_i}) + d(t_{B_i}) + d(t_{C_i})), \end{aligned} \tag{4.4}$$

since $q(B_i) > (3/7)\lambda$, $q(C_i) \geq (5/7)\lambda$, $q(A_i \cup B_i \cup C_i) \geq (12/7)\lambda$, and $d(t_{A_i}) \leq d(t_{B_i}) \leq d(t_{C_i})$ hold.

Hence the proof completes by summing inequalities (4.2), (4.3), and (4.4) overall subsets in $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$. □

We now turn to proving that a solution output from algorithm ApproxMCEI is within a factor of $(15/8 + \rho_{\mathrm{ST}})$ of the optimal solution.

Theorem 4.1. *For an instance $I = (G, w, \lambda, s, M, q)$ of* MCEI*, algorithm* ApproxMCEI *delivers a $(15/8 + \rho_{\mathrm{ST}})$-approximate solution $\mathcal{P}$.*

Proof. The algorithm first produces a tree T of minimum cost including all vertices in $M \cup \{s\}$. For each source $t \in M' = \{t \in M \mid q(t) > \lambda/2\}$, we install a copy of each edge in a shortest path between s and t in (G, w). We then find a partition $\mathcal{M} = \{Z_0\} \cup \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3$ of the set $M - M'$ of the remaining sources, and assign a hub vertex t_Z for each subset $Z \in \mathcal{M}$, such that when the total demand in each subset is routed to its hub vertex simultaneously, the amount of these flow on each edge of T is at most λ. Finally, for each set $Z \in \mathcal{M}$, we install a copy of each edge in a shortest path between s and t_Z in (G, w) ($t_{Z_0} = s$). Hence, the cost of the constructed set $\mathcal{P}$ of paths is bounded by

$$cost(\mathcal{P}) \leq w(T) + \sum_{t \in M'} d(t) + \sum_{Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z).$$

For a minimum Steiner tree T^* that spans $M \cup \{s\}$ and a weight $opt(I)$ of an optimal solution, we have $w(T) \leq \rho_{\mathrm{ST}} w(T^*)$ and $w(T^*) \leq opt(I)$ by Lemma 4.1. Hence $w(T) \leq \rho_{\mathrm{ST}} \cdot opt(I)$ holds. To prove the theorem, it suffices to show that

$$\sum_{t \in M'} d(t) + \sum_{Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z) \leq (15/8) opt(I). \tag{4.5}$$

To prove this inequality, we distinguish two different cases (i) and (ii).

(i) $\sum_{t \in M'} q(t)d(t) \geq \sum_{t \in M - M'} q(t)d(t)$: Then Lemma 4.1 implies that

$$\begin{aligned} opt(I) &\geq (1/\lambda) \sum_{t \in M} q(t)d(t) = (1/\lambda)\Big(\sum_{t \in M'} q(t)d(t) + \sum_{t \in M - M'} q(t)d(t)\Big) \\ &\geq (2/\lambda) \sum_{t \in M - M'} q(t)d(t) \geq (2/\lambda) \sum_{t \in Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} q(t)d(t) \\ &\geq (8/7) \sum_{Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z), \end{aligned} \tag{4.6}$$

where the last inequality follows from Lemma 4.8. Inequality (4.6) and Lemma 4.2 prove (4.5) in this case.

(ii) $\sum_{t \in M'} q(t)d(t) < \sum_{t \in M - M'} q(t)d(t)$: Then it is easy to see that there exist two nonnegative real numbers α and β such that $\alpha + \beta = 1$, $\alpha < \beta$, $(1/\lambda) \sum_{t \in M'} q(t)d(t) \leq \alpha opt(I)$, and $(1/\lambda) \sum_{t \in M - M'} q(t)d(t) \leq \beta opt(I)$. Since $q(t) > \lambda/2$ for all $t \in M'$, we have

$$(1/2) \sum_{t \in M'} d(t) < (1/\lambda) \sum_{t \in M'} q(t)d(t) \leq \alpha opt(I). \tag{4.7}$$

On the other hand, Lemma 4.8 implies that

$$(4/7) \sum_{Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z) \leq (1/\lambda) \sum_{t \in Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} q(t)d(t) \leq \beta opt(I). \tag{4.8}$$

By multiplying (4.7) and (4.8) by 2 and 7/4, respectively, and adding the obtained inequalities, we have

$$\sum_{t \in M'} d(t) + \sum_{Z \in \mathcal{M}_1 \cup \mathcal{M}_2 \cup \mathcal{M}_3} d(t_Z) \leq (2\alpha + (7/4)\beta) opt(I) < (15/8) opt(I),$$

by the assumptions on α and β. □

Chapter 5

The Capacitated Tree-Routing Problem in Networks

In this chapter, we study the capacitated tree-routing problem, in which we are asked to construct a minimum cost set of trees of the network each of which rooted at a fixed vertex and has a limited demand. Each edge has a capacity which stands for the maximum number of the constructed trees allowed to contain this edge. Note that CMTR studied in Chapter 2 is equivalent to instances of this problem with unit edge capacities. Moreover, CND defined in Chapter 4 is equivalent to this problem when each terminal has a unit demand.

5.1 Introduction

We propose an extension model of routing problems in networks which includes a set of important routing problems as special cases. This extension generalizes two different routing protocols in networks, where we model the underlying network using an undirected edge-weighted graph, a set of terminals with positive demands, and a designated vertex s. In the first protocol defined in CMTR, we are given a network with routing capacity $\kappa > 0$, and we are interested in finding a set $\mathcal{T}$ of trees rooted at s each of which contains terminals whose total demand does not exceed κ. The objective is to minimize the total weight of all trees in $\mathcal{T}$, where the weight of a tree is the sum of edge weights among all edges in the tree. In the other protocol defined in CND, we are given a network with an edge capacity λ, and we are interested in finding a set $\mathcal{P}$ of paths P_v between each terminal v and s, where the demand of v should be routed via P_v to s. A subset of paths of $\mathcal{P}$ can contain an edge in the underlying network as long as the total demand of the paths in this subset does not exceed the capacity λ. For any edge e, any integer number of copies of e can be installed. The goal is to minimize the total weight of all edges installed in the network.

These routing protocols play an important role in many applications such as communication networks supporting multimedia and the design of telecommunication and transportation

networks. Our new problem formulation can be applied to possible extensions of these applications.

One possible application can be found in a video delivery system in a computer network. We are given a graph $G = (V, E)$ with a set V of nodes, a set E of links, a cost function $w : E \to R^+$, and a link bandwidth $\lambda > 0$. We have a service center $s \in V$ and a set $M \subseteq V$ of clients (terminals) with demands $q : M \to R^+$. The service center s actually consists of a large number of servers, each of which can serve at most κ demands from clients that are assigned to it. Notice that, if we use IP multicast, then for each server and its clients, the routing subgraph connecting them must be a tree (see [73] for the detail). Suppose that we can install as many links as possible. Then the problem is to find an assignment of clients to servers that minimizes the total link installation cost without violating the capacity of every server and the bandwidth of every link, where the latter is considered as the traffic due to the routing, i.e., the number of servers using the link. For such a problem, no approximation algorithm has been obtained.

In this chapter, we consider a capacitated routing problem under a multi-tree model. Under this model, we are interested in constructing a set $\mathcal{T}$ of tree-routings that connects given terminals to a sink s in a network with a routing capacity $\kappa > 0$ and an edge capacity $\lambda > 0$. A network is modeled with an edge-weighted undirected graph. Each terminal has a demand > 0, and a tree in the graph can connect s and a subset of terminals whose total demand does not exceed the routing capacity κ. The weight of an edge in a network stands for the cost of installing a copy of the edge. A subset of trees can pass through a single copy of an edge e as long as the number of these trees does not exceed the edge capacity λ; any integer number of copies of e are allowed to be installed. The goal is to find a feasible set of tree-routings that minimizes the total weight of edges installed in the network. We call this problem the *capacitated tree-routing problem* (CTR for short), which can be formally stated as follows.

Capacitated Tree-Routing Problem (CTR):
Input: A connected graph $G = (V, E)$, an edge weight function $w : E \to R^+$, a routing capacity $\kappa > 0$, an integer edge capacity $\lambda \geq 1$, a sink $s \in V$, a set $M \subseteq V - \{s\}$ of terminals, and a demand function $q : M \to R^+$.
Feasible solution: A partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_\ell\}$ of M and a set $\mathcal{T} = \{T_1, T_2, \ldots, T_\ell\}$ of trees of G such that $Z_i \cup \{s\} \subseteq V(T_i)$ and $q(Z_i) \leq \kappa$ hold for each i. The number of copies of an edge $e \in E$ installed in the solution is given by $h_{\mathcal{T}}(e) = \lceil |\{T \in \mathcal{T} \mid e \in E(T)\}|/\lambda \rceil$.
Goal: Minimize the sum of weights of edges to be installed under the edge capacity constraint, that is,

$$\sum_{e \in E} h_{\mathcal{T}}(e) w(e).$$

CTR is our new problem formulation which includes several important routing problems

as its special cases. First of all, CTR with $\lambda = 1$ and $\kappa \geq \sum_{v \in M} q(v)$ is equivalent to the Steiner tree problem. Secondly CTR is closely related to CND and CMTR. In particular, CTR and CND are equivalent in the case where $\kappa = 1$ and $q(v) = 1$ for every $v \in M$, and CMTR is equivalent to CTR with $\lambda = 1$. Therefore, CTR is a considerably general model for routing problems. In this chapter, we prove that CTR admits a $(2 + \rho_{\mathrm{ST}})$-approximation algorithm. For this, we derive a new result on tree covers in graphs.

The rest of this section introduces two lower bounds on $opt(I)$.

Lemma 5.1. *Let $I = (G, w, \kappa, \lambda, s, M, q)$ be a* CTR *instance. Then it hold*

$$w(T^*) \leq opt(I) \text{ for a minimum cost Steiner tree } T^* \text{ to } (G, w, M \cup \{s\}), \tag{5.1}$$

$$1/(\kappa\lambda) \sum_{v \in M} d_{(G,w)}(s, v) q(v) \leq opt(I). \tag{5.2}$$

Proof. Let $(\mathcal{M}^* = \{Z_1, \ldots, Z_\ell\}, \mathcal{T}^* = \{T_1, \ldots, T_\ell\})$ be an optimal solution to CTR instance I. Hence $opt(I) = \sum_{e \in E} h_{\mathcal{T}^*}(e) w(e)$.

We first show (5.1). Let $E(\mathcal{T}^*) = \cup_{T_i \in \mathcal{T}^*} E(T_i)$ $(\subseteq E(G))$, i.e., the set of all edges used in the optimal solution. Then the edge set $E(\mathcal{T}^*)$ contains a tree T that spans $M \cup \{s\}$ in G. We see that the cost $w(T)$ of T in G is at most that of the CTR solution, proving (5.1).

We next show (5.2). Since $|\{T_i \in \mathcal{T}^* \mid e \in E(T_i)\}| \leq \lambda h_{\mathcal{T}^*}(e)$ holds for all $e \in E$, we see that

$$\sum_{T_i \in \mathcal{T}^*} w(T_i) \leq \sum_{e \in E} \lambda h_{\mathcal{T}^*}(e) w(e) = \lambda opt(I). \tag{5.3}$$

On the other hand, for each tree $T_i \in \mathcal{T}^*$, we have

$$\sum_{v \in Z_i} q(v) d_{(G,w)}(s, v) \leq w(T_i) q(Z_i) \leq \kappa w(T_i), \tag{5.4}$$

since $w(T_i) \geq d_{(G,w)}(s, v)$ for all $v \in V(T_i)$. Hence by summing (5.4) overall trees in $\mathcal{T}^*$ and using (5.3), we obtain (5.2). □

As mentioned above, CMTR is equivalent to CTR with $\lambda = 1$. Thus CTR instances with $\lambda = 1$ are $(2+\rho_{\mathrm{ST}})$-approximable by the results on CMTR described in Chapter 2. It remains to approximate CTR instances with $\lambda \geq 2$. In the subsequent sections we use the algorithm described for CMTR in Section 2.2, and show how it can be modified to approximate CTR.

5.2 Simple approximation algorithm

In this section we show that CTR is $(4 + \rho_{\mathrm{ST}})$-approximable by using simple results on tree covers in a tree. We first recall the algorithm designed for the general demand case of CMTR.

Algorithm ApproxCTR
Input: A CTR instance $I = (G, w, \kappa, \lambda \geq 2, s, M, q)$.
Output: A solution $(\mathcal{M}, \mathcal{T})$ to I.

Step 1. Compute a ρ_{ST}-approximate solution T to the Steiner tree problem in (G, w) that spans $M \cup \{s\}$ and then regard T as a tree rooted at s.
Define a vertex weight function $d : M \to R^+$ by setting

$$d(v) := d_{(G,w)}(s, v), \quad v \in M.$$

Step 2. Find a partition $\mathcal{M}$ of M.
For each subset $Z \in \mathcal{M}$, assign a vertex $t_Z \in V(T)$ as its hub vertex.
Let S be the set of all hub vertices.

Step 3. For each hub vertex $t \in S$, we choose a shortest path $SP(s, t)$ between s and t in (G, w). For each subset $Z \in \mathcal{M}$, let T_Z be the tree obtained from $T\langle Z \cup \{t_Z\}\rangle$ by adding the edge set in $SP(s, t_Z)$. Let $\mathcal{T} := \{T_Z \mid Z \in \mathcal{M}\}$. □

For $\lambda \geq 2$, a straightforward generalization can be obtained by realizing Step 2 of ApproxCTR so that more than one subtree can share a hub vertex in the following way. After Step 1 is performed, we construct a partition $\mathcal{M} = \cup_{1 \leq j \leq g} \mathcal{C}_j$ of M such that each $\mathcal{C}_j$ consists of a κ-balanced partition of a subtree of T in Step 2, where a common hub vertex t_j will be assigned to all subsets in $\mathcal{C}_j$.

More formally, we apply a procedure that

- first chooses a vertex v with the maximum depth in the current tree such that $q(V(T_v) \cap M) > \kappa\lambda/4$,
- selects all subsets in a κ-balanced partition of the terminal set in T_v to form the next collection $\mathcal{C}_j$, and
- chooses a hub vertex t_j with the minimum distance $d(t_j)$ among the terminals $\mathcal{C}_j$ before removing all terminals in $\mathcal{C}_j$ from M.

We repeat the procedure on the minimal subtree of T that contains the current M and s as long as $q(M) > \kappa\lambda/4$ holds. Finally, find a κ-balanced partition of the remaining tree, and let the subsets in the partition form $\mathcal{C}_g$, setting $t_g = s$. Let $S = \{t_j \mid j = 1, 2, \ldots, g\}$, and $t_Z := t_j$ for each $Z \in \mathcal{M}$ and j with $Z \in \mathcal{C}_j$.

Note that, for each $j = 1, 2, \ldots, g$, it holds $\sum_{Z \in \mathcal{C}_j} q(Z) \leq \kappa\lambda/2$ by the choice of v, and hence Property (ii) of κ-balanced partition implies that $|\mathcal{C}_j| < \lambda$. It is easy to verify that the obtained partition $\mathcal{M} = \cup_{1 \leq j \leq g} \mathcal{C}_j$ satisfies the following lemma.

Lemma 5.2. *For a rooted tree T rooted at s, a terminal set $M \subseteq V(T) - \{s\}$, a demand function $q : M \to R^+$, a real κ with $\kappa \geq \max\{q(v) \mid v \in M\}$, and an integer $\lambda \geq 2$, a partition $\mathcal{M} = \cup_{1 \leq j \leq g} \mathcal{C}_j$ of M computed above satisfies:*

(i) *$q(Z) \leq \kappa$ for all $Z \in \mathcal{M}$, and $T\langle Z\rangle$ and $T\langle Z'\rangle$ have no common edge for all distinct $Z, Z' \in \mathcal{M}$;*

(ii) $|\mathcal{C}_j| < \lambda$ *for all* $j = 1, 2, \ldots, g$, *and* $\sum_{Z \in \mathcal{C}_j} q(Z) > \kappa\lambda/4$ *for all* $j = 1, 2, \ldots, g-1$; *and*

(iii) *Each* $T\langle \cup_{Z \in \mathcal{C}_i} Z \rangle$ *and* $T\langle \cup_{Z \in \mathcal{C}_j} Z \rangle$ *have no common edge for* $i \neq j$. □

Then we perform Step 3 of ApproxCTR. For the resulting tree-routings $\mathcal{T} = \{T_Z \mid Z \in \mathcal{M}\}$, the installing cost satisfies

$$\sum_{e \in E} h_{\mathcal{T}}(e) w(e) \leq w(T) + \sum_{t \in S} d(t), \tag{5.5}$$

by the edge-disjointness in Lemma 5.2(i) and (iii). Hence it suffices to show that $\sum_{t \in S} d(t) \leq 4opt(I)$. By the choice of hub vertices t_j and since $\sum_{Z \in \mathcal{C}_j} q(Z) > \kappa\lambda/4$ for all $j = 1, 2, \ldots, g-1$, we conclude that

$$(1/4) \sum_{t \in S} d(t) < \sum_{1 \leq j \leq g-1} \sum_{v \in Z \in \mathcal{C}_j} q(v) d(v)/(\kappa\lambda) \leq \sum_{v \in M} q(v) d(v)/(\kappa\lambda).$$

Hence, (5.2) in Lemma 5.1 implies that the above algorithm delivers a $(4 + \rho_{\mathrm{ST}})$-approximate solution $(\mathcal{M}, \mathcal{T})$ for CTR with $\lambda \geq 2$.

In the next section, we improve the ratio to $(2 + \rho_{\mathrm{ST}})$ by using new tree covers and a swapping method for CND problem due to [33].

5.3 Improved approximation algorithm

This section shows that ApproxCTR with an additional step, Step 4, can deliver a $(2+\rho_{\mathrm{ST}})$-approximate solution for a CTR instance with $\lambda \geq 2$. For this, we use the following result on tree covers in a tree to realize Step 2, where a proof of the lemma will be given in Section 5.4.

For a partition $\mathcal{M}$ of a terminal set M in a rooted tree T and hub vertices t_Z, $Z \in \mathcal{M}$, we denote the set of subsets $Z \in \mathcal{M}$ such that $T\langle Z \cup \{t_Z\} \rangle$ contains a specified edge $e = (x, y) \in E(T)$ with $y \in Ch_T(x)$ by three disjoint sets:

$\mathcal{M}(e) = \{Z \in \mathcal{M} \mid e \in E(T\langle Z \rangle)\}$,

$\mathcal{M}_{dwn}(e) = \{Z \in \mathcal{M} \mid Z \subseteq V(T) - V(T_y), t_Z \in V(T_y)\}$, and

$\mathcal{M}_{up}(e) = \{Z \in \mathcal{M} \mid Z \subseteq V(T_y), t_Z \in V(T) - V(T_y)\}$.

Consider a tree T described in Fig. 5.1(a). Note that $Z_5, Z'_4, Z'_5 \subseteq V(T_y)$ and Z_1, Z_2, Z_3, Z_4, $Z'_1, Z'_2, Z'_3 \subseteq V(T) - V(T_y)$. Moreover, all subsets in $\{Z_1, \ldots, Z_5\}$ and $\{Z'_1, \ldots, Z'_5\}$ are assigned to $t_j \in V(T_y)$ and $t_{j'} \in V(T) - V(T_y)$, respectively. This implies that $\mathcal{M}(e) = \emptyset$, $\mathcal{M}_{dwn}(e) = \{Z_1, Z_2, Z_3, Z_4\}$, and $\mathcal{M}_{up}(e) = \{Z'_4, Z'_5\}$.

Lemma 5.3. *Let* T *be a tree rooted at* s *with a terminal set* $M \subseteq V(T) - \{s\}$, *a demand function* $q : M \to R^+$, *a real* κ *with* $\kappa \geq \max\{q(v) \mid v \in M\}$, *and an integer* $\lambda \geq 2$. *Given a vertex weight function* $d : M \to R^+$, *there exist a partition* $\mathcal{M} = \cup_{1 \leq j \leq f} \mathcal{C}_j$ *of* M, *and a set* $S = \{t_j \in \{\mathrm{argmin}_{t \in Z \in \mathcal{C}_j} d(t)\} \mid j \leq f-1\} \cup \{t_f = s\}$ *of hub vertices such that:*

(i) $q(Z) \leq \kappa$ *for all* $Z \in \mathcal{M}$, *and* $T\langle Z\rangle$ *and* $T\langle Z'\rangle$ *have no common edge for all distinct* $Z, Z' \in \mathcal{M}$;

(ii) $|\mathcal{C}_j| \leq \lambda$ *for all* $j = 1, 2, \ldots, f$, *and* $\sum_{Z\in\mathcal{C}_j} q(Z) > \kappa\lambda/2$ *for all* $j = 1, 2, \ldots, f-1$; *and*

(iii) *For* $t_Z = t_j$ *with* $Z \in \mathcal{C}_j$, $j = 1, 2, \ldots, f$, *each edge* $e \in E(T)$ *satisfies*
(a) $|\mathcal{M}(e)| \leq 1$,
(b) $|\mathcal{M}_{dwn}(e)| \leq \lambda - 1$, *and*
(c) $|\mathcal{M}_{up}(e)| \leq \lambda - 1$.

Furthermore, a pair $(\mathcal{M}, S)$ *can be computed in polynomial time.* □

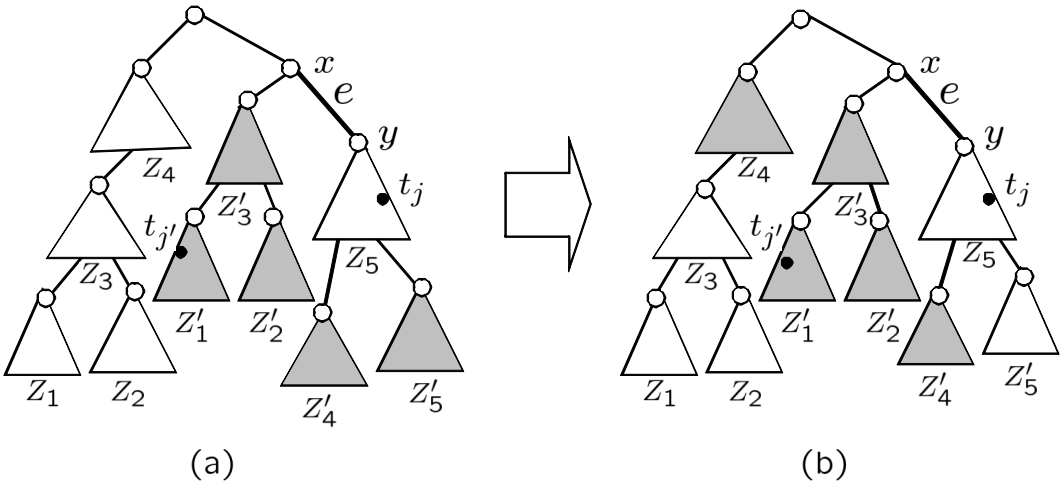

Figure 5.1: Illustration of swapping process, where Z_4 and Z'_5 are swapped between $\mathcal{M}_{dwn}(e)$ and $\mathcal{M}_{up}(e)$; (a) $\mathcal{M}_{dwn}(e) = \{Z_1, Z_2, Z_3, Z_4\}$ and $\mathcal{M}_{up}(e) = \{Z'_4, Z'_5\}$; (b) $\mathcal{M}_{dwn}(e) = \{Z_1, Z_2, Z_3\}$ and $\mathcal{M}_{up}(e) = \{Z'_4\}$.

To construct a $(2 + \rho_{\mathrm{ST}})$ approximate solution to a given CTR instance $I = (G, w, \kappa, \lambda \geq 2, s, M, q)$, we first perform Step 1 of ApproxCTR.

In Step 2, we apply Lemma 5.3 to the Steiner tree T and the function d obtained in Step 1 to get a partition $\mathcal{M} = \cup_{1\leq j\leq f}\mathcal{C}_j$ of M and a set $S = \{t_1, t_2, \ldots, t_f\}$ of hub vertices that satisfy conditions of the lemma, and we set $t_Z = t_j$ for each $Z \in \mathcal{C}_j$, $j = 1, 2, \ldots, f$.

Then we perform Step 3 for the set $\mathcal{T}' = \{T\langle Z \cup \{t_Z\}\rangle \mid Z \in \mathcal{M}\}$ of induced subtrees of T. Note that each collection $\mathcal{C}_j$, $j = 1, 2, \ldots, f$, contains at most λ subsets from $\mathcal{M}$, all of which can use t_j as a common hub vertex by installing one copy of each edge in $SP(s, t_j)$. We here analyze the installing cost of the resulting tree-routing. Analogously with the previous section, we have $\sum_{t\in S} d(t) \leq 2opt(I)$, since it holds by Lemma 5.3(i)-(ii) that

$$(1/2)\sum_{t\in S} d(t) < \sum_{1\leq j\leq f-1} \sum_{v\in Z\in\mathcal{C}_j} q(v)d(v)/(\kappa\lambda) \leq \sum_{v\in M} q(v)d(v)/(\kappa\lambda).$$

It should be noted that an edge $e \in E(T)$ may be used more than λ times in the subtrees in $\mathcal{T}'$, and (5.5) may not hold for the current tree-routing.

Finally we perform Step 4, an additional step, in order to modify the assignment of hub vertices so that (5.5) holds, which implies the $(2+\rho_{\mathrm{ST}})$-approximability of CTR. Consider an edge $e=(x,y)$ in the Steiner tree T, where by definition the number of trees in $\mathcal{T}'$ containing e equals $|\mathcal{M}_{dwn}(e)|+|\mathcal{M}_{up}(e)|+|\mathcal{M}(e)|$. Assume that the total number of trees in $\mathcal{T}'$ containing e exceeds λ; i.e.,

$$|\mathcal{M}_{dwn}(e)|+|\mathcal{M}_{up}(e)|+|\mathcal{M}(e)|>\lambda,$$

which implies

$$\mathcal{M}_{dwn}(e)\neq\emptyset \text{ and } \mathcal{M}_{up}(e)\neq\emptyset$$

by Lemma 5.3(iii)(a)-(c). In this case, we choose two subsets $Z\in\mathcal{M}_{dwn}(e)$ and $Z'\in\mathcal{M}_{up}(e)$, where Z and Z' belong to distinct collections $\mathcal{C}_j$ and $\mathcal{C}_{j'}$, respectively, and swap them; i.e., we reassign the hub vertices of Z and Z' such that $t_Z:=t_{j'}$ and $t_{Z'}:=t_j$. As a result, $\mathcal{M}_{dwn}(e)$, $\mathcal{M}_{up}(e)$, $\mathcal{C}_j$, and $\mathcal{C}_{j'}$ are updated such that $\mathcal{M}_{dwn}(e):=\mathcal{M}_{dwn}(e)-\{Z\}$, $\mathcal{M}_{up}(e):=\mathcal{M}_{up}(e)-\{Z'\}$, $\mathcal{C}_j:=(\mathcal{C}_j-\{Z\})\cup\{Z'\}$, and $\mathcal{C}_{j'}:=(\mathcal{C}_{j'}-\{Z'\})\cup\{Z\}$. Also, $\mathcal{T}'$ is updated accordingly such that

$$\mathcal{T}':=\mathcal{T}'-\{T\langle Z\cup\{t_j\}\rangle,T\langle Z'\cup\{t_{j'}\}\rangle\}\cup\{T\langle Z\cup\{t_Z\}\rangle+T\langle Z'\cup\{t_{Z'}\}\rangle\}.$$

The swapping operation decreases the number of trees in $\mathcal{T}'$ containing each edge in $E(T\langle Z\cup\{t_j\}\rangle)\cap E(T\langle Z'\cup\{t_{j'}\}\rangle)$ (which includes e), where t_j and $t_{j'}$ were the previous hub vertices of Z and Z', respectively, and hence $|\mathcal{M}_{dwn}(e)|,|\mathcal{M}_{up}(e)|\leq\lambda-1$ still holds. Note that the number of trees in $\mathcal{T}'$ containing each of the remaining edges of T never increases. We repeat the swapping process as long as the number of trees in the current $\mathcal{T}'$ containing the edge e exceeds λ.

Step 4 repeats the process for any edge of T shared by more than λ trees of the current $\mathcal{T}'$. Step 4 never changes the set S of hub vertices computed in Lemma 5.3. Therefore, the set $\mathcal{T}=\{T_Z\mid Z\in\mathcal{M}\}$ of tree-routings T_Z obtained from each tree $T\langle Z\cup\{t_Z\}\rangle$ of the final set $\mathcal{T}'$ by adding the edge set of $SP(s,t_Z)$ satisfies (5.5) and is a $(2+\rho_{\mathrm{ST}})$-approximate solution to the given CTR instance I.

Fig. 5.1 illustrates of swapping process in a CTR instance with $\lambda=5$. In Fig. 5.1(a) $\mathcal{C}_j=\{Z_1,\ldots,Z_5\}$, $\mathcal{C}_{j'}=\{Z'_1,\ldots,Z'_5\}$, $\mathcal{M}_{dwn}(e)=\{Z_1,Z_2,Z_3,Z_4\}$, $\mathcal{M}_{up}(e)=\{Z'_4,Z'_5\}$, and t_j and $t_{j'}$ are the hub vertices of $\mathcal{C}_j$ and $\mathcal{C}_{j'}$, respectively. The number of trees of $\mathcal{T}'=\{T\langle Z\cup\{t_Z\}\rangle\mid Z\in\mathcal{M}\}$ containing e equals 6 which exceeds λ. In Fig. 5.1(b) Z_4 and Z'_5 are swapped between $\mathcal{C}_j$ and $\mathcal{C}_{j'}$ so that $\mathcal{C}_j:=(\mathcal{C}_j-\{Z_4\})\cup\{Z'_5\}$ and $\mathcal{C}_{j'}:=(\mathcal{C}_{j'}-\{Z'_5\})\cup\{Z_4\}$. Moreover, $\mathcal{T}'$ is updated so that $\mathcal{T}':=\big(\mathcal{T}'-\{T\langle Z_4\cup\{t_j\}\rangle,T\langle Z'_5\cup\{t_{j'}\}\rangle\}\big)\cup\{T\langle Z'_5\cup\{t_j\}\rangle,T\langle Z_4\cup\{t_{j'}\}\rangle\}$. The number of trees of the updated $\mathcal{T}'$ containing e is reduced to 4 which is less than λ.

Hence we have the following theorem.

Theorem 5.1. *CTR with $\lambda\geq 2$ is $(2+\rho_{\mathrm{ST}})$-approximable.*

5.4 Proof of Lemma 5.3

The purpose of this section is to provide a proof of Lemma 5.3. We first assume for simplicity and without loss of generality that the given tree T is binary and $M = L(T)$.

We prove Lemma 5.3 by showing that algorithm TreeCover actually delivers a desired pair $(\mathcal{M}, S)$. The algorithm constructs collections $\mathcal{C}_1, \mathcal{C}_2, \ldots,$ by applying a procedure that first chooses a vertex v with the maximum depth in the current tree such that $q(V(T_v) \cap M) > \kappa\lambda/2$, finds κ-balanced partitions of terminal sets of T_u and $T_{\tilde{u}}$ of the two children u and $\tilde{u}$ of v, and then selects several subsets in the obtained partitions to form the next new collection $\mathcal{C}_j$ (see Fig. 5.2(a)). We then remove all terminals in $\mathcal{C}_j$ from M and repeat this procedure on the minimal subtree of T that contains the current set M and s as long as $q(M) > \kappa\lambda/2$ holds. Finally, let a κ-balanced partition of the current set M form the last collection $\mathcal{C}_f$.

Algorithm TreeCover
Input: A binary tree $\widehat{T}$ rooted at s, a terminal set $M = L(\widehat{T})$, a demand function $q : M \to R^+$, a real κ with $\kappa \geq \max\{q(v) \mid v \in M\}$, an integer $\lambda \geq 2$, and a vertex weight function $d : M \to R^+$.
Output: A pair $(\mathcal{M}, S)$ that satisfies Conditions (i)-(iii) in Lemma 5.3.
Initialize: $T := \widehat{T}$ and $j := 0$.

1. **While** The current T has a vertex v with $q(V(T_v) \cap M) \geq \kappa\lambda/2$ **do**
2. $j := j + 1$; Choose such v with the maximum depth in T;
3. Denote $Ch_T(v) := \{u, \tilde{u}\}$; $Z_t := V(T_t) \cap M$ for $t \in \{u, \tilde{u}\}$;
4. Find κ-balanced partitions $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_p\}$ and $\widetilde{\mathcal{Z}} = \{\widetilde{Z}_1, \widetilde{Z}_2, \ldots, \widetilde{Z}_{\tilde{p}}\}$ of Z_u and $Z_{\tilde{u}}$, respectively (see Fig. 5.2(a));
5. Let t_j be the terminal of the smallest vertex weight d in $V(T_v)$, where we assume that $t_j \in V(T_u)$ w.o.l.g;
6. Let $\mathcal{C}_j$ consist of all subsets of $\mathcal{Z}$ and a minimal family $\{\widetilde{Z}_1, \widetilde{Z}_2, \ldots, \widetilde{Z}_b\}$, $b \leq \tilde{p}$, of subsets of $\widetilde{\mathcal{Z}}$ so that $\sum_{Z \in \mathcal{C}_j} q(Z) > \kappa\lambda/2$;
7. $M := M - \cup_{Z \in \mathcal{C}_j} Z$; $T := T\langle M \cup \{s\}\rangle$
   /* $t_j \notin V(T)$ */
8. **endwhile**;
   /* $q(M) < \kappa\lambda/2$ */
9. $f := j + 1$; $t_f := s$;
10. **if** $M = \emptyset$ **then**
11. $\mathcal{C}_f := \emptyset$
12. **else**
    Let $\mathcal{C}_f$ be a κ-balanced partition of M
13. **endif**;

14. For each $j = 1, 2, \ldots, f$ and $Z \in \mathcal{C}_j$, let $t_Z := t_j$;
15. Let $\mathcal{M} := \cup_{1 \le j \le f} \mathcal{C}_j$ and $S := \{t_j \mid 1 \le j \le f\}$.

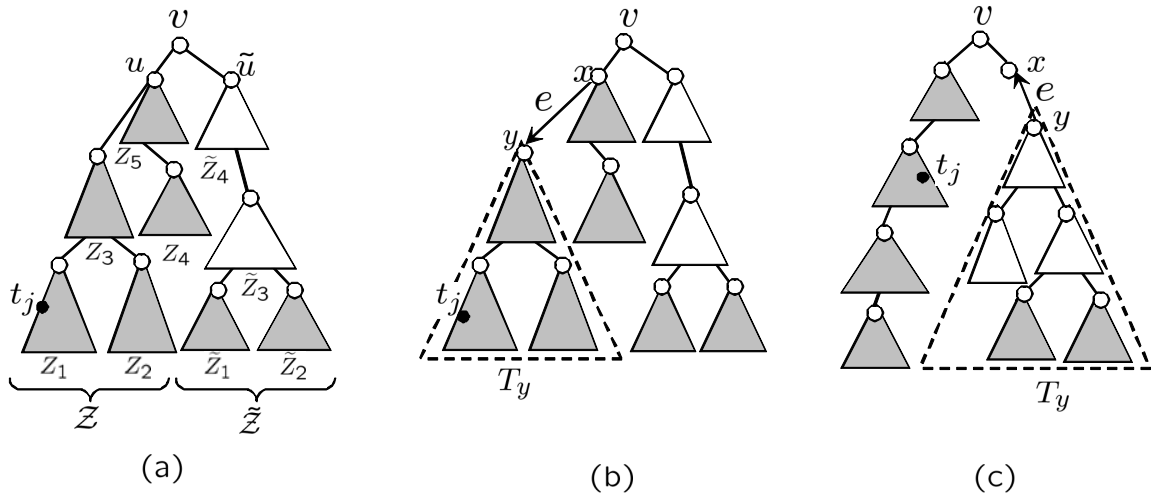

Figure 5.2: (a) the construction of a collection $\mathcal{C}_j$ (gray triangles) computed in line 6 of algorithm TreeCover, where t_j is the terminal of the minimum distance d in T_v; (b) illustration for Lemma 5.3(iii)(b); (c) illustration for Lemma 5.3(iii)(c).

Consider the first moment a subset in $\mathcal{M}_{dwn}(e)$ is assigned to a vertex in $V(T_y)$. Then $y \in V(T_u)$ and $t_j \in V(T_y)$. Moreover, no vertices in $V(T_u)$ will be a hub vertex again since T_u will be removed from T. Also, Fig. 5.2(c) illustrates Lemma 5.3(iii)(c). Consider the first moment a subset in $\mathcal{M}_{up}(e)$ is assigned to a vertex in $V(T) - V(T_y)$. The number of subsets in $\mathcal{M}_{up}(e)$ is bounded by the number of subsets with terminals in $T_{\tilde{u}}$.

Now we prove that the output $(\mathcal{M}, S)$ of algorithm TreeCover satisfies conditions (i)-(iii) in Lemma 5.3.

(i) Note that $\mathcal{C}_j \subseteq \mathcal{Z} \cup \widetilde{\mathcal{Z}}$ holds for any collection $\mathcal{C}_j$ computed in line 6, where $\mathcal{Z}$ and $\widetilde{\mathcal{Z}}$ are the κ-balanced partitions computed in line 4. Therefore, by Property (i) of κ-balanced partition in Section 3.3, each subset of $\mathcal{C}_j$ has at most κ demand. Similarly, we see that each subset of $\mathcal{C}_f$ computed in line 11 has at most κ demand.

Now we prove the second part of (i). Consider the execution of the jth iteration of the algorithm. By the construction of $\mathcal{C}_j$ and Property (iii) of κ-balanced partition, we have $E(T\langle Z\rangle) \cap E(T\langle Z'\rangle) = \emptyset$ for any distinct $Z, Z' \in \mathcal{C}_j$, where T is the current tree during the jth iteration. Moreover, the same property implies that $E(T\langle Z\rangle) \cap E(T\langle (M - \cup_{Z \in \mathcal{C}_j} Z) \cup \{s\}\rangle) = \emptyset$ for all $Z \in \mathcal{C}_j$. Hence for any distinct subsets $Z, Z' \in \mathcal{M}$, we have $E(\widehat{T}\langle Z\rangle) \cap E(\widehat{T}\langle Z'\rangle) = \emptyset$ since a partition $\mathcal{M}$ of M output by TreeCover is a union of collections $\mathcal{C}_j$, $j = 1, 2, \ldots, f$. This completes the proof of (i).

(ii) Consider a collection $\mathcal{C}_j$ computed in line 6. From Property (ii) of κ-balanced partition, we have $q(Z_{p-1} \cup Z_p) > \kappa$ and $q(\widetilde{Z}_{\tilde{p}-1} \cup Z_{\tilde{p}}) > \kappa$ in the partitions $\mathcal{Z}$ and $\widetilde{\mathcal{Z}}$ computed in line 4. Moreover, $q(Z_i) > \kappa/2$ (resp., $q(\widetilde{Z}_i) > \kappa/2$) for all $i = 1, 2, \ldots, p-1$ (resp., $i = 1, 2, \ldots, \tilde{p}-1$). Hence the minimality of subsets of $\mathcal{C}_j$ chosen from $\widetilde{\mathcal{Z}}$ implies that $|\mathcal{C}_j| \leq \lambda$. For a collection $\mathcal{C}_f$ computed in line 11, Property (ii) of the κ-balanced partition of the current set M implies that $|\mathcal{C}_f| \leq \lambda$ since $q(M) \leq \kappa\lambda/2$. This proves (ii)

Finally we prove condition (iii)(a)-(c).

(a) From condition (i), we have $E(\widehat{T}\langle Z\rangle) \cap E(\widehat{T}\langle Z'\rangle) = \emptyset$ for all distinct subsets $Z, Z' \in \mathcal{M}$. This means that there exists at most one subset $Z \in \mathcal{M}$ such that $e \in E(\widehat{T}\langle Z\rangle)$ and consequently $|\mathcal{M}(e)| \leq 1$, which proves (a).

Note that throughout processing of any subtree T_v (for a vertex v chosen in line 2), the algorithm does not assign any subset in $\{Z \in \mathcal{M} \mid Z \subseteq V(\widehat{T}) - V(\widehat{T}_v)\}$ (resp., $\{Z \in \mathcal{M} \mid Z \subseteq V(\widehat{T}_v)\}$) to a hub vertex in $V(\widehat{T}_v)$ (resp., $V(\widehat{T}) - V(\widehat{T}_v)$). This implies that when $y \notin D_{\widehat{T}}(v) - v$, none of the subsets in $\{Z \in \mathcal{M} \mid Z \subseteq V(\widehat{T}) - V(\widehat{T}_y)\}$ (resp., $\{Z \in \mathcal{M} \mid Z \subseteq V(\widehat{T}_y)\}$) is assigned to a hub vertex in $V(\widehat{T}_y)$ (resp., $V(\widehat{T}) - V(\widehat{T}_y)$). Then it is sufficient to prove properties (b) and (c) for the subtree $T = \widehat{T}\langle(M - \cup_{i<j}(\cup_{Z\in\mathcal{C}_i} Z)) \cup \{s\}\rangle$, where the vertex v chosen in line 2 of the jth iteration must satisfy $y \in D_T(v) - \{v\}$.

(b) Consider the first moment when a subset in $\mathcal{M}_{dwn}(e)$ is assigned to a hub vertex in $V(T_y)$ during the execution of TreeCover. Let v be the vertex such that the tree T_v with $y \in D_T(v) - \{v\}$ is being processed in the jth iteration of the algorithm. The algorithm first chooses a vertex $t_j \in V(T_v)$ in line 5, where $t_j \in V(T_u)$ is assumed without loss of generality, and then constructs a collection $\mathcal{C}_j$ such that all terminals in $V(T_u)$ are contained in $\mathcal{C}_j$ (since $\mathcal{Z} \subseteq \mathcal{C}_j$) and all subsets of $\mathcal{C}_j$ are assigned to a hub vertex t_j (see Fig. 5.2(a)). This implies that $y \in V(T_u)$ and $t_j \in V(T_y)$. Moreover, once a set of subsets in $\mathcal{M}_{dwn}(e)$ is assigned to a hub vertex in T_y in an iteration of the algorithm, none of the vertices of T_y will become a hub vertex in the subsequent iterations since all terminals in $\mathcal{C}_j$ (and hence in $V(T_u)$) will be removed from the terminal set in the next iterations (see line 7). Therefore, all subsets in $\mathcal{M}_{dwn}(e)$ assigned to a hub vertex in $V(T_y)$ are assigned to t_j in this iteration. On the other hand, the number of subsets assigned to t_j (which equals $|\mathcal{C}_j|$) is the sum of the number of subsets in $\mathcal{M}_{dwn}(e)$ and subsets in $\{Z \in \mathcal{M} \mid Z \cap V(T_y) \neq \emptyset\}$. There exists at least one subset in the latter set since $t_j \in V(T_y)$. Hence the number of subsets in $\mathcal{M}_{dwn}(e)$ is at most $\lambda - 1$ since $|\mathcal{C}_j| \leq \lambda$. This proves (b).

(c) Consider the first moment when a subset in $\mathcal{M}_{up}(e)$ is assigned to a hub vertex in $V(T) - V(T_y)$ during the execution of TreeCover. Let v be the vertex such that the tree T_v with $y \in D_T(v) - \{v\}$ is being processed in the jth iteration of the algorithm. Note that $|\mathcal{Z}| < \lambda$ holds in a κ-balanced partition $\mathcal{Z}$ of Z_u computed in line 4 since otherwise $q(V(T_u) \cap M) > \kappa\lambda/2$ would violate the choice of v (by $\lambda \geq 2$). Similarly, $|\widetilde{\mathcal{Z}}| < \lambda$ holds in a partition $\widetilde{\mathcal{Z}}$ of $Z_{\tilde{u}}$ computed in the same line. Hence, $|\{Z \in \mathcal{M} \mid Z \cap V(T_u) \neq \emptyset\}| < \lambda$ and $|\{Z \in \mathcal{M} \mid Z \cap V(T_{\tilde{u}}) \neq \emptyset\}| < \lambda$ hold (since $\mathcal{M} = \cup_{1\leq j\leq f}\mathcal{C}_j$). This implies that

$|\{Z \in \mathcal{M} \mid Z \cap V(T_y) \neq \emptyset\}| < \lambda$ since $y \in D_T(u)$ or $y \in D_T(\tilde{u})$ holds. That is, the number of subsets in $\mathcal{M}_{up}(e)$ is at most $\lambda - 1$. This proves (c). □

Chapter 6

The Generalized Capacitated Tree-routing Problem

In this chapter, we introduce the generalized capacitated tree-routing problem which is described as follows. Given a connected graph $G = (V, E)$ with a sink $s \in V$ and a set $M \subseteq V - \{s\}$ of terminals with a nonnegative demand $q(v)$, $v \in M$, we wish to find a collection of trees rooted at s to send all the demands to s, where the total demand collected by each tree is bounded from above by a demand capacity $\kappa > 0$. Let $\lambda > 0$ denote a bulk capacity of an edge, and each edge $e \in E$ has an installation cost $w(e) \geq 0$ per bulk capacity; each edge e is allowed to have capacity $j\lambda$ for any integer j, which installation incurs cost $jw(e)$. To establish a desired tree routing T_i, each edge e contained in T_i requires $\alpha + \beta q'$ amount of capacity for the total demand q' that passes through edge e along T_i, where $\alpha \geq 0$ and $\beta \geq 0$ are prescribed constants. Term α means a fixed amount used to separate the inside of the routing T_i from the outside while term $\beta q'$ means the net capacity proportional to q'. The objective of GCTR is to find a collection of trees that minimizes the total installation cost of edges. GCTR is a new generalization which unifies several known routing problems in networks with edge/demand capacities.

6.1 Introduction

In this chapter, we introduce the *generalized capacitated tree-routing problem* (GCTR), which is described as follows. Given a connected graph $G = (V, E)$ with a demand capacity $\kappa > 0$, a bulk edge capacity $\lambda > 0$, a sink $s \in V$, and a set $M \subseteq V - \{s\}$ of terminals with a nonnegative demand $q(v)$, $v \in M$, we wish to find a collection $\mathcal{T} = \{T_1, T_2, \ldots, T_\ell\}$ of trees rooted at s to send all the demands to s, where the total demand in the set Z_i of terminals assigned to tree T_i does not exceed the demand capacity κ. Each edge $e \in E$ has an installation cost $w(e) \geq 0$ per bulk capacity; each edge e is allowed to have capacity $j\lambda$ for any integer j, which requires installation cost $jw(e)$. To establish a tree routing T_i through an edge e, we

assume that e needs to have capacity at least

$$\alpha + \beta q(Z_i \cap D_{T_i}(v_i^e))$$

for prescribed coefficients $\alpha, \beta \geq 0$, where v_i^e is the tail of e in T_i; α means a fixed amount used to separate the inside and outside of the routing T_i while term $\beta q(Z_i \cap D_{T_i}(v_i^e))$ means the net capacity proportional to the amount $q(Z_i \cap D_{T_i}(v_i^e))$ of demands that passes through edge e along T_i. Hence, given a set $\mathcal{T} = \{T_1, T_2, \ldots, T_\ell\}$ of trees, each edge e needs to have capacity $h_{\mathcal{T}}(e)\lambda$ for the least integer $h_{\mathcal{T}}(e)$ such that

$$\sum_{T_i \in \mathcal{T}: T_i \text{ contains } e} \left(\alpha + \beta q(Z_i \cap D_{T_i}(v_i^e))\right) \leq h_{\mathcal{T}}(e)\lambda,$$

and the total installation cost of edges incurred by $\mathcal{T}$ is given as $\sum_{e \in E} h_{\mathcal{T}}(e)w(e)$, where $h_{\mathcal{T}}(e) = 0$ if no $T_i \in \mathcal{T}$ contains e. The objective of GCTR is to find a set $\mathcal{T}$ of trees that minimizes the total installation cost of edges. We formally state GCTR as follows.

Generalized Capacitated Tree-Routing Problem (GCTR):
Input: A connected graph $G = (V, E)$, an edge weight function $w : E \to R^+$, a demand capacity $\kappa > 0$, an edge capacity $\lambda > 0$, prescribed constants $\alpha, \beta \geq 0$, a sink $s \in V$, a set $M \subseteq V - \{s\}$ of terminals, and a demand function $q : M \to R^+$.
Feasible solution: A partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_\ell\}$ of M and a set $\mathcal{T} = \{T_1, T_2, \ldots, T_\ell\}$ of trees of G such that $Z_i \cup \{s\} \subseteq V(T_i)$ and $q(Z_i) \leq \kappa$ hold for each i. The number of copies of an edge $e \in E$ installed in the solution is given by $h_{\mathcal{T}}(e) = \lceil \sum_{T_i : e \in E(T_i)} (\alpha + \beta q(Z_i \cap D_{T_i}(v_i^e)))/\lambda \rceil$, where v_i^e is the tail of e in T_i.
Goal: Minimize the total installation cost of $\mathcal{T}$, that is,

$$\sum_{e \in E} h_{\mathcal{T}}(e)w(e).$$

We have a variant of GCTR if it is allowed to purchase edge capacity in any required quantity. In this model, for each edge e of the underlying network, we assign capacity of $\lambda_e = \alpha|\mathcal{T}'| + \beta \sum_{T_i \in \mathcal{T}'} q(Z_i \cap D_{T_i}(v_i^e))$ on e, where $\mathcal{T}'$ is the set of trees containing e. That is, the total cost of the constructed trees equals $\sum_{e \in E} \lambda_e w(e)$. We call this variant of GCTR, *the fractional generalized capacitated tree-routing problem* (FGCTR).

We easily see that GCTR and FGCTR contain two classical NP-hard problems, the Steiner tree problem and the *bin packing problem* [22]. We see that GCTR with an edge weighted graph G, $\alpha = \lambda = 1$, and $\beta = 0$ is equivalent to the Steiner tree problem in G when $\kappa \geq \sum_{v \in M} q(v)$, whereas it is equivalent to the bin packing problem with bin size κ when G is a complete graph, $w(e) = 1$ for all edges e incident to s and $w(e) = 0$ otherwise. We see that FGCTR also has a similar relationship with the Steiner tree problem and the bin packing problem.

The characteristic of GCTR and FGCTR is their routing capacity which is a linear combination of the number of trees and the total amount of demands that pass through an edge. Such a general form of capacity constraint can be found in some applications.

Suppose that we wish to find a minimum number of trucks to carry given n items $v_1, v_2, \ldots, v_n$, where each item v_i has size $q(v_i)$ and weight $\beta q(v_i)$, where β is a specific gravity. We also have bins; the weight of a bin is α and the capacity of a bin is κ. Items are first put into several bins, and then the bins are assigned to trucks under capacity constraints. That is, we can put items in a bin B so that the total size $\sum_{v_i \in B} q(v_i)$ of the items does not exceed the bin capacity κ, where the weight of the bin B is given by a linear combination $\alpha + \beta \sum_{v_i \in B} q(v_i)$. We can load packed bins into a truck as long as the total weight of these packed bins does not exceed the truck capacity λ. The objective is to find assignments of items to bins and packed bins to trucks such that the number of required trucks is minimized. This problem can be described as GCTR.

Suppose that a petroleum corporation wishes to construct a network of pipelines to collect raw oil from several locations to a set of storage stations (to be specified among all locations), each of which has a specified demand capacity, and then send the oil from these storage stations to a specified major refinery. Moreover, for the sake of efficiency, the corporation staff wants to construct a set of trees that spans all locations, each of which contains a storage station. A single pipe type with a specified bulk capacity is available. For each edge of the underlying pipe network, it is allowed to install either zero or an integer number of pipes, where each pipe has a nonnegative construction cost. A part of pipe capacity is used to protect the internal surface of the pipe, while the rest of the pipe capacity needs to be proportional to the amount of oil that goes through the pipe. Therefore, the required amount of capacity of edge is given as a linear combination of the number of trees that and the total demand pass through the edge. The goal of the corporation is to construct the cheapest possible set of feasible tree-routings so that the demands of all locations can be routed simultaneously to the refinery without violating the capacity constraint.

Another application can be found in a generalized model of the video delivery system in computer science discussed in Section 5.1 such that the objective is to find an assignment of clients to servers that minimizes the total link installation cost without violating the capacity of every server and the bandwidth of every link, where the latter is considered as a linear combination of the traffic due to the routing (the number of servers using the link) and the data communication (the total data going through the link).

Similar routing problems in which the objective function is a linear combination of two or more optimization requirements have been studied before [6, 7, 69]. For example, given a *lattice graph* with an edge capacity and a vertex cost function, *the global routing problem in VLSI design* asks to construct a set of trees that spans a given set of *nets* (subsets of the vertex set) under an edge capacity constraint. Terlaky et al. [69] have studied a problem of minimizing an objective function which is defined as a linear combination of the total edge

cost and the total number of bends of all trees, where a bend at a vertex corresponds a via in VLSI design, which leads to extra cost in manufacturing.

We here observe that our new problem formulation, GCTR, includes several important routing problems as its special cases. Note that GCTR with $\alpha = 1$ and $\beta = 0$ is equivalent to CTR proposed in Chapter 5. This implies that GCTR with $\alpha = 0$, $\beta = 1$, and $\kappa = \lambda$ is equivalent to CND. Also, GCTR with $\alpha = 1$, $\beta = 0$, and $\lambda = 1$ is equivalent to CMTR.

As observed above, GCTR is a considerably general model for routing problems. In this chapter, we first prove that GCTR admits a $(2\lceil\lambda/(\alpha+\beta\kappa)\rceil/\lfloor\lambda/(\alpha+\beta\kappa)\rfloor+\rho_{\mathrm{ST}})$-approximation algorithm if $\lambda \geq \alpha+\beta\kappa$ holds. The high-level description of the proposed algorithm resembles our algorithm for CTR discussed in the previous chapter, but we need to derive a new lower bound to the problem. Namely, given an instance $I = (G, w, \kappa, \lambda, \alpha, \beta, s, M, q)$ of GCTR, the main idea of our algorithm is to compute an integer capacity λ' depending on κ, λ, α, and β and then find a feasible tree-routings solution to the instance $I' = (G, w, \kappa, \lambda', s, M, q)$ of CTR. Here such a capacity λ' is chosen so that this set of tree-routings is a feasible solution to the original GCTR instance I.

We observe that it is not straightforward to modify the above algorithm so that it also delivers a constant-factor approximate solution in the case of $\lambda < \alpha + \beta\kappa$. This motivates proposing a different approach for approximating GCTR instances with $\lambda < \alpha+\beta\kappa$. For this, we introduce a new lower bound on GCTR by introducing a generalization of CND, and use a balanced Steiner tree as a base tree from which we construct a collection of trees to send demands to sink. We show that our new algorithm delivers a 13.037-approximate solution to GCTR with $\lambda < \alpha + \beta\kappa$. Based on the same approach, we also prove that FGCTR is 8.529-approximable.

Table 6.1 shows a summary of the recent approximation algorithms for CND, CMTR, CTR, and GCTR. Note that $\theta = \lceil\lambda/(\alpha + \beta\kappa)\rceil/\lfloor\lambda/(\alpha + \beta\kappa)\rfloor$ is less than 2.

6.2 Preliminaries

This section introduces two lower bounds on the optimal value to GCTR. The first lower bound is based on the Steiner tree problem.

Lemma 6.1. *Given a* GCTR *instance* $I = (G, w, \kappa, \lambda, \alpha, \beta, s, M, q)$*, the minimum cost of a Steiner tree to* $(G, w, M \cup \{s\})$ *is a lower bound on the optimal value to* GCTR *instance* I*.*

Proof. Consider an optimal solution $(\mathcal{M}^* = \{Z_1, \ldots, Z_\ell\}, \mathcal{T}^* = \{T_1, \ldots, T_\ell\})$ to GCTR instance I. Let $E(\mathcal{T}^*) = \cup_{T_i \in \mathcal{T}^*} E(T_i)$ $(\subseteq E(G))$, i.e., the set of all edges used in the optimal solution. Then the edge set $E(\mathcal{T}^*)$ contains a tree T that spans $M \cup \{s\}$ in G. We see that the cost $w(T)$ of T in G is at most that of GCTR solution. Hence the minimum cost of a Steiner tree to $(G, w, M \cup \{s\})$ is no more than the optimal value to GCTR instance I. □

Table 6.1: Approximation algorithms for CND, CMTR, CTR, and GCTR problems, where $\theta = \lceil \lambda/(\alpha+\beta\kappa) \rceil / \lfloor \lambda/(\alpha+\beta\kappa) \rfloor$.

Problem		unit demands $q \equiv 1$	general demands $q \geq 0$
CND	$\alpha = 0, \beta = 1,$ $\kappa = \lambda \in R^+$	$1 + \rho_{\text{ST}}$ [33]	$2 + \rho_{\text{ST}}$ [33]
CMTR	$\alpha = 1, \beta = 0,$ $\lambda = 1, \kappa \in R^+$	$8/5 + (5/4)\rho_{\text{ST}}$ [11], $3/2 + (4/3)\rho_{\text{ST}}$ [56] (Chapter 2)	$2 + \rho_{\text{ST}}$ [40] (Chapter 2)
CTR	$\alpha = 1, \beta = 0$ $\lambda \in Z^+$, $\kappa \in R^+$	$2 + \rho_{\text{ST}}$ [55] (Chapter 5)	$2 + \rho_{\text{ST}}$ [55] (Chapter 5)
GCTR	$\alpha, \beta, \kappa, \lambda \in R^+$ with (i) $\lambda \geq \alpha + \beta\kappa$ (ii) $\lambda < \alpha + \beta\kappa$	$2\theta + \rho_{\text{ST}}$ [57] (this chapter) 13.037 (this chapter)	$2\theta + \rho_{\text{ST}}$ [57] (this chapter) 13.037 (this chapter)

The second lower bound is derived from an observation on the distance from vertices to sink s.

Lemma 6.2. *Let $I = (G, w, \kappa, \lambda, \alpha, \beta, s, M, q)$ be an instance of* GCTR*. Then*

$$(\alpha + \beta\kappa)/(\kappa\lambda) \sum_{v \in M} q(v) d_{(G,w)}(s, v)$$

is a lower bound on the optimal value to GCTR *instance I.*

Proof. Consider an optimal solution $(\mathcal{M}^* = \{Z_1, \ldots, Z_\ell\}, \mathcal{T}^* = \{T_1, \ldots, T_\ell\})$ to GCTR instance I. For each edge $e \in E(T_i)$, $i = 1, 2, \ldots, \ell$, we assume that $e = (u_i^e, v_i^e)$, where $v_i^e \in Ch_{T_i}(u_i^e)$. Let $opt(I)$ denote the optimal value of GCTR instance I. Then we have

$$\begin{aligned}
opt(I) &= \sum_{e \in E} \left\lceil [\alpha | \{T_i \mid e \in E(T_i)\}| + \beta \sum_{T_i : e \in E(T_i)} q(Z_i \cap D_{T_i}(v_i^e))]/\lambda \right\rceil w(e) \\
&\geq \sum_{e \in E} w(e)[\alpha | \{T_i \mid e \in E(T_i)\}| + \beta \sum_{T_i : e \in E(T_i)} q(Z_i \cap D_{T_i}(v_i^e))]/\lambda \\
&= (\alpha/\lambda) \sum_{e \in E} |\{T_i \mid e \in E(T_i)\}| w(e) \\
&\quad + (\beta/\lambda) \sum_{e \in E} w(e) \sum_{T_i : e \in E(T_i)} q(Z_i \cap D_{T_i}(v_i^e)) \\
&= (\alpha/\lambda) \sum_{T_i \in \mathcal{T}^*} w(T_i) + (\beta/\lambda) \sum_{T_i \in \mathcal{T}^*} \sum_{e \in E(T_i)} q(Z_i \cap D_{T_i}(v_i^e)) w(e). \qquad (6.1)
\end{aligned}$$

Note that, for each tree $T_i \in \mathcal{T}^*$, we have

$$\kappa w(T_i) \geq w(T_i) \sum_{v \in Z_i} q(v) \geq \sum_{v \in Z_i} q(v) d_{(G,w)}(s, v), \qquad (6.2)$$

since $w(T_i) \geq d_{(G,w)}(s,v)$ for all $v \in V(T_i)$. On the other hand, for each tree $T_i \in \mathcal{T}^*$, we have

$$\begin{aligned} \sum_{e \in E(T_i)} q(Z_i \cap D_{T_i}(v_i^e))w(e) &= \sum_{v \in Z_i} q(v) d_{(T_i,w)}(s,v) \\ &\geq \sum_{v \in Z_i} q(v) d_{(G,w)}(s,v). \end{aligned} \tag{6.3}$$

Hence by summing (6.2) and (6.3) overall trees in $\mathcal{T}^*$ and substituting in (6.1), we conclude that

$$(\alpha + \beta\kappa)/(\lambda\kappa) \sum_{v \in M} q(v) d_{(G,w)}(s,v) \leq opt(I),$$

which completes the proof. □

6.3 Approximation algorithm for $\lambda \geq \alpha + \beta\kappa$

In this section we present an approximation algorithm to GCTR instances with $\lambda \geq \alpha + \beta\kappa$.

Given an instance $I = (G, w, \kappa, \lambda, \alpha, \beta, s, M, q)$ of GCTR, the main idea of our algorithm is to find a feasible solution $(\mathcal{M} = \{Z_1, \ldots, Z_\ell\}, \mathcal{T} = \{T_1, \ldots, T_\ell\})$ to a CTR instance $I' = (G, w, \kappa, \lambda', s, M, q)$, where $\lambda' = \lfloor \lambda/(\alpha + \beta\kappa) \rfloor$. That is, for each edge e in G, the number of trees of $\mathcal{T}$ containing e is at most $h_{\mathcal{T}}(e)\lambda'$, where $h_{\mathcal{T}}(e)$ denotes the number of copies of e installed in the solution $(\mathcal{M}, \mathcal{T})$ of I'. Note that $q(Z_i) \leq \kappa$ for all $i = 1, 2, \ldots, \ell$. Therefore, for each edge e in G with tail v^e, we have

$$\begin{aligned} \sum_{T_i \in \mathcal{T}: e \in E(T_i)} (\alpha + \beta q(D_{T_i}(v^e) \cap M)) &\leq (\alpha + \beta\kappa) |\{T_i \in \mathcal{T} \mid e \in E(T_i)\}| \\ &\leq h_{\mathcal{T}}(e)(\alpha + \beta\kappa) \lfloor \lambda/(\alpha + \beta\kappa) \rfloor \leq h_{\mathcal{T}}(e)\lambda. \end{aligned}$$

This implies that $(\mathcal{M}, \mathcal{T})$ is a feasible solution to GCTR instance I.

For seeking a simple presentation, we first discuss GCTR instances with $\lfloor \lambda/(\alpha+\beta\kappa) \rfloor = 1$ in the next section.

6.3.1 Approximation algorithm for $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor = 1$

This section provides an approximate solution to GCTR when $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor = 1$. The algorithm is based on κ-balanced partition. For convenience, we first recall the definition of κ-balanced partition.

For a tree T rooted at a vertex r, an ordered partition $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_p\}$ of a subset of the terminal set M is called *κ-balanced* if the following holds:

(i) $q(Z_i) \leq \kappa$ for $i = 1, 2, \ldots, p$;

(ii) $q(Z_i) > \kappa/2$ for $i = 1, 2, \ldots, p-1$, and if $p \geq 2$ then $q(Z_{p-1} \cup Z_p) > \kappa$; and

(iii) Each $T\langle Z_j \rangle$ $(j = 1, 2, \ldots, p-1)$ has no common edge with $T\langle \cup_{j<i\leq p} Z_i + r \rangle$.

We proved in Chapter 2 that such a κ-balanced partition always exists if $\max_{v\in M} q(v) \leq \kappa$ (see Lemma 2.2).

The basic idea of the algorithm is analogous to that for CTR given in the previous chapter. We first compute an approximate Steiner tree T in $(G, w, M \cup \{s\})$, regard T as a tree rooted at s, and then find a κ-balanced partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_p\}$ of M in T. For each $Z_i \in \mathcal{M}$, we choose a vertex $t_{Z_i} \in Z_i$ and connect the tree $T\langle Z_i \rangle$ to s by adding a shortest path between s and t_{Z_i} in (G, w). We describe the algorithm in the following form which will be used for the case of $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor \geq 2$.

Algorithm ApproxGCTR
Input: A GCTR instance $I = (G, w, \kappa, \lambda, \alpha, \beta, s, M, q)$.
Output: A solution $(\mathcal{M}, \mathcal{T})$ to I.

Step 1. Compute a ρ_{ST}-approximate solution T to the Steiner tree problem in (G, w) that spans $M \cup \{s\}$ and then regard T as a tree rooted at s.
Define a vertex weight function $d : M \to R^+$ by setting

$$d(v) := d_{(G,w)}(s, v), \quad v \in M.$$

Step 2. Find a partition $\mathcal{M}$ of M.
For each subset $Z \in \mathcal{M}$, assign a vertex $t_Z \in V(T)$ as its hub vertex.
Let S be the set of all hub vertices.

Step 3. For each hub vertex $t \in S$, we choose a shortest path $SP(s, t)$ between s and t in (G, w). For each subset $Z \in \mathcal{M}$, let T_Z be the tree obtained from $T\langle Z \cup \{t_Z\} \rangle$ by adding the edge set in $SP(s, t_Z)$. Let $\mathcal{T} := \{T_Z \mid Z \in \mathcal{M}\}$. □

For a GCTR instance with $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor = 1$, we realize Step 2 as follows. We compute a κ-balanced partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_p\}$ of M. For $j = 1, 2, \ldots, p-1$, we choose a terminal $t_{Z_j} \in Z_j$ with the minimum distance $d(t_{Z_j})$ as its hub vertex, and let $t_{Z_p} := s$ for $j = p$.

Theorem 6.1. *Given a* GCTR *instance with* $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor = 1$*, algorithm* ApproxGCTR *with the above Step 2 delivers a* $(2\lambda/(\alpha + \beta\kappa) + \rho_{\mathrm{ST}})$*-approximate solution.*

Proof. By Property (iii) of κ-balanced partition, each edge in T is used at most once in the union of subtrees in $\mathcal{T}' = \{T\langle Z_j \rangle \mid j = 1, 2, \ldots, p-1\} \cup \{T\langle Z_p \cup \{s\} \rangle\}$. Furthermore, the flow on each edge in T is at most $\alpha + \beta\kappa \leq \lambda$. On the other hand, the flow on each edge in $SP(s, t_{Z_i})$, $i = 1, 2, \ldots, p-1$, is at most $\alpha + \beta\kappa \leq \lambda$. Note that $\mathcal{T}' = \{T\langle Z_i \cup \{t_{Z_i}\} \rangle \mid Z_i \in \mathcal{M}\}$ by the choice of hub vertices. Therefore, $(\mathcal{M}, \mathcal{T})$ is feasible and the total weight of the edges

to be installed for $\mathcal{T}$ is bounded by the weight of T plus the sum of the shortest paths used; i.e., it holds

$$\sum_{e\in E} h_{\mathcal{T}}(e)w(e) \le w(T) + \sum_{1\le i\le p-1} d(t_{Z_i}). \tag{6.4}$$

For a minimum Steiner tree T^* that spans $M\cup\{s\}$, we have $w(T^*)\le opt(I)$ by Lemma 6.1. Hence $w(T)\le \rho_{\mathrm{ST}}\cdot w(T^*)\le \rho_{\mathrm{ST}}\cdot opt(I)$ holds. To prove the theorem, it suffices to show that

$$\sum_{1\le i\le p-1} d(t_{Z_i}) \le 2\lambda/(\alpha+\beta\kappa)opt(I). \tag{6.5}$$

The choice of hub vertices and Property (ii) of κ-balanced partition imply that, for each $i=1,2,\ldots,p-1$, we have

$$\sum_{v\in Z_i} q(v)d(v) \ge d(t_{Z_i})\sum_{v\in Z_i} q(v) > d(t_{Z_i})\kappa/2. \tag{6.6}$$

By summing inequality (6.6) overall $i=1,2,\ldots,p-1$, we have

$$\begin{aligned}(\alpha+\beta\kappa)/(2\lambda)\sum_{1\le i\le p-1} d(t_{Z_i}) &< (\alpha+\beta\kappa)/(\kappa\lambda)\sum_{1\le i\le p-1}\sum_{v\in Z_i} q(v)d(v)\\ &\le (\alpha+\beta\kappa)/(\kappa\lambda)\sum_{t\in M} q(t)d(t).\end{aligned}$$

By Lemma 6.2, this proves (6.5). □

6.3.2 Approximation algorithm for $\lfloor\lambda/(\alpha+\beta\kappa)\rfloor\ge 2$

This section shows that ApproxGCTR with an additional step, Step 4, can deliver a $([2\lambda/(\alpha+\beta\kappa)]/\lfloor\lambda/(\alpha+\beta\kappa)\rfloor+\rho_{\mathrm{ST}})$-approximate solution for a GCTR instance with $\lfloor\lambda/(\alpha+\beta\kappa)\rfloor\ge 2$. For this, we use the following result on tree covers in a tree to realize Step 2. The result is the same as Lemma 5.3 by replacing λ with $\lfloor\lambda/(\alpha+\beta\kappa)\rfloor$. We state the lemma here for completeness.

Lemma 6.3. *Let T be a tree rooted at s with a terminal set $M\subseteq V(T)-\{s\}$, a demand function $q: M\to R^+$, a real κ with $\kappa\ge\max\{q(v)\mid v\in M\}$, a real $\lambda>0$, and real constants $\alpha,\beta\ge 0$. Given a vertex weight function $d: M\to R^+$, there exist a partition $\mathcal{M}=\cup_{1\le j\le f}\mathcal{C}_j$ of M, and a set $S=\{t_j\in\{\mathrm{argmin}_{t\in Z\in\mathcal{C}_j} d(t)\}\mid j\le f-1\}\cup\{t_f=s\}$ of hub vertices such that:*

(i) *$q(Z)\le\kappa$ for all $Z\in\mathcal{M}$, and $T\langle Z\rangle$ and $T\langle Z'\rangle$ have no common edge for all distinct $Z,Z'\in\mathcal{M}$;*

(ii) *$|\mathcal{C}_j|\le\lfloor\lambda/(\alpha+\beta\kappa)\rfloor$ for all $j=1,2,\ldots,f$, and $\sum_{Z\in\mathcal{C}_j} q(Z)>\lfloor\lambda/(\alpha+\beta\kappa)\rfloor(\kappa/2)$ for all $j=1,2,\ldots,f-1$; and*

(iii) *For $t_Z = t_j$ with $Z \in \mathcal{C}_j$, $j = 1, 2, \ldots, f$, each edge $e \in E(T)$ satisfies*
  (a) $|\mathcal{M}(e)| \leq 1$,
  (b) $|\mathcal{M}_{dwn}(e)| \leq \lfloor \lambda/(\alpha + \beta\kappa) \rfloor - 1$, *and*
  (c) $|\mathcal{M}_{up}(e)| \leq \lfloor \lambda/(\alpha + \beta\kappa) \rfloor - 1$.

Here $\mathcal{M}(e)$, $\mathcal{M}_{dwn}(e)$, and $\mathcal{M}_{up}(e)$ are defined as in Section 5.3. □

We first perform Step 1 of ApproxGCTR. In Step 2, we apply Lemma 6.3 to the Steiner tree T and the function d obtained in Step 1 to get a partition $\mathcal{M} = \cup_{1 \leq j \leq f} \mathcal{C}_j$ of M and a set $S = \{t_1, t_2, \ldots, t_f\}$ of hub vertices that satisfy the conditions of Lemma 6.3, and we set $t_Z = t_j$ for each $Z \in \mathcal{C}_j$, $j = 1, 2, \ldots, f$. Then we perform Step 3 for the set $\mathcal{T}' = \{T\langle Z \cup \{t_Z\}\rangle \mid Z \in \mathcal{M}\}$ of induced subtrees of T. Note that each collection $\mathcal{C}_j$, $j = 1, 2, \ldots, f$, contains at most $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor$ subsets from $\mathcal{M}$, all of which can use t_j as a common hub vertex by installing one copy of each edge in $SP(s, t_j)$. We here analyze the installing cost of the resulting tree-routing. Analogously with the previous section, we have

$$\sum_{1 \leq j \leq f-1} d(t_j) \leq [2\lambda/(\alpha + \beta\kappa)]/\lfloor \lambda/(\alpha + \beta\kappa) \rfloor opt(I),$$

since it holds by Lemma 6.3(i)-(ii) that

$$\begin{aligned}(\alpha + \beta\kappa)\lfloor \lambda/(\alpha + \beta\kappa) \rfloor/(2\lambda) \sum_{1 \leq j \leq f-1} d(t_j) &< (\alpha + \beta\kappa)/(\kappa\lambda) \sum_{1 \leq j \leq f-1} \sum_{t \in Z \in \mathcal{C}_j} q(t)d(t) \\ &\leq (\alpha + \beta\kappa)/(\kappa\lambda) \sum_{t \in M} q(t)d(t).\end{aligned}$$

It should be noted that the flow on an edge $e \in E(T)$ may be more than λ and (6.4) may not hold for the current tree-routing.

Finally we perform Step 4 in order to modify the assignment of hub vertices so that (6.4) holds, which implies the $([2\lambda/(\alpha + \beta\kappa)]/\lfloor \lambda/(\alpha + \beta\kappa) \rfloor + \rho_{\mathrm{ST}})$-approximability of GCTR with $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor \geq 2$. Consider an edge $e = (x, y)$ in the Steiner tree T, where by definition the number of trees in $\mathcal{T}'$ containing e equals $|\mathcal{M}_{dwn}(e)| + |\mathcal{M}_{up}(e)| + |\mathcal{M}(e)|$. Assume that the total number of trees in $\mathcal{T}'$ containing e exceeds $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor$; i.e.,

$$|\mathcal{M}_{dwn}(e)| + |\mathcal{M}_{up}(e)| + |\mathcal{M}(e)| > \lfloor \lambda/(\alpha + \beta\kappa) \rfloor,$$

which implies

$$|\{T' \in \mathcal{T}' \mid e \in E(T')\}| > \lfloor \lambda/(\alpha + \beta\kappa) \rfloor.$$

Step 4 repeats a swapping process for any edge of T shared by more than $\lfloor \lambda/(\alpha + \beta\kappa) \rfloor$ trees of the current $\mathcal{T}'$. See Chapter 5 (Section 5.3) for the details of such a swapping process. Step 4 never changes the set S of hub vertices computed in Lemma 6.3.

Therefore, the set $\mathcal{T} = \{T_Z \mid Z \in \mathcal{M}\}$ of tree-routings T_Z obtained from each tree $T\langle Z \cup \{t_Z\}\rangle$ of $\mathcal{T}'$ by adding the edge set of $SP(s, t_Z)$ satisfies (6.4) and is a $([2\lambda/(\alpha + \beta\kappa)]/\lfloor \lambda/(\alpha + \beta\kappa) \rfloor + \rho_{\mathrm{ST}})$-approximate solution to the given GCTR instance I. Hence we have the following theorem.

Theorem 6.2. GCTR *with* $\lfloor \lambda/(\alpha+\beta\kappa) \rfloor \geq 2$ *is* $([2\lambda/(\alpha+\beta\kappa)]/\lfloor \lambda/(\alpha+\beta\kappa) \rfloor + \rho_{\mathrm{ST}})$-*approximable.*

6.4 Approximation algorithm for $\lambda < \alpha + \beta\kappa$

As we mentioned before, it is not straightforward to modify the algorithm in the previous section so that it also delivers a constant-factor approximate solution in the case of $\lambda < \alpha+\beta\kappa$. In this section, we introduce a new lower bound on GCTR by introducing a generalization of CND in Section 6.4.1, and use a balanced Steiner tree as a base tree from which we construct a collection of trees to send demands to sink. We prove an approximation algorithm of 13.037 for the problem in this case.

The following lemma introduces another lower bound to GCTR based on the Steiner tree problem which is equivalent to that given in Lemma 6.1 for a GCTR instance with $\alpha \leq \lambda$.

Lemma 6.4. *Let* $I = (G, w, \kappa, \lambda, \alpha, \beta, s, M, q)$ *be an instance of* GCTR *and* T^* *be a minimum cost Steiner tree to* $(G, w, M \cup \{s\})$. *Then* $\lceil \alpha/\lambda \rceil w(T^*)$ *is a lower bound on the optimal value to* I.

Proof. Consider an optimal solution $(\mathcal{M}^* = \{Z_1, \ldots, Z_\ell\}, \mathcal{T}^* = \{T_1, \ldots, T_\ell\})$ to I with optimal value $opt(I)$. For each edge $e \in E(T_i)$, $i = 1, 2, \ldots, \ell$, we assume that $e = (u_i^e, v_i^e)$, where $v_i^e \in Ch_{T_i}(u_i^e)$. Let $E(\mathcal{T}^*) = \cup_{T_i \in \mathcal{T}^*} E(T_i)$ $(\subseteq E(G))$, i.e., the set of all edges used in the optimal solution. Then

$$\begin{aligned} opt(I) &= \sum_{e \in E(\mathcal{T}^*)} \lceil \sum_{T_i : e \in E(T_i)} (\alpha + \beta q(Z_i \cap D_{T_i}(v_i^e)))/\lambda \rceil w(e) \\ &\geq \lceil \alpha/\lambda \rceil \sum_{e \in E(\mathcal{T}^*)} w(e) \geq \lceil \alpha/\lambda \rceil \sum_{e \in E(T^*)} w(e), \end{aligned}$$

since the edge set $E(\mathcal{T}^*)$ contains a tree that spans $M \cup \{s\}$ in G. □

6.4.1 Generalized capacitated network design problem

In this section, we propose a generalized version of CND, *the generalized capacitated network design problem* (GCND), which defines a new lower bound to the optimal value of GCTR. We show that such a lower bound can be used to construct a constant factor approximation algorithm to GCTR instances with $\lambda < \alpha + \beta\kappa$. We are given a graph $G = (V, E)$ with a bulk edge capacity $\lambda > 0$, a sink $s \in V$, and a set $M \subseteq V - \{s\}$ of terminals with a nonnegative demand $q(v)$, $v \in M$. The problem asks to choose a path P_v from each terminal $v \in M$ to the sink along which the demand $q(v)$ of v is sent to s. Each edge $e \in E$ has an installation cost $w(e) \geq 0$ per bulk capacity; each edge e is allowed to have capacity $j\lambda$ for any integer j, which requires installation cost $jw(e)$. Hence, given a set $\mathcal{P} = \{P_v \mid v \in M\}$ of paths of G,

each edge e in $E(\mathcal{P}) = \cup_{v \in M} E(P_v)$ needs to have capacity $k_{\mathcal{P}}(e)\lambda$ for the least integer $k_{\mathcal{P}}(e)$ such that

$$\alpha + \beta \sum_{v \in M : P_v \text{ contains } e} q(v) \leq k_{\mathcal{P}}(e)\lambda,$$

where $k_{\mathcal{P}}(e) = 0$ if no path contains e. The total installation cost of edges incurred by $\mathcal{P}$ is given as $\sum_{e \in E(\mathcal{P})} k_{\mathcal{P}}(e)w(e)$. The objective of GCND is to minimizes the total installation cost of edges. The problem is formally stated as follows.

Generalized Capacitated Network Design Problem (GCND):
Input: A connected graph $G = (V, E)$, an edge weight function $w : E \to R^+$, an edge capacity $\lambda > 0$, and prescribed constants $\alpha, \beta \geq 0$, a sink $s \in V$, a set $M \subseteq V - \{s\}$ of terminals, and a demand function $q : M \to R^+$.
Feasible solution: A set $\mathcal{P} = \{P_v \mid v \in M\}$ of paths of G such that $\{s, v\} \subseteq V(P_v)$ holds for each $v \in M$. The number of copies of an edge e in $E(\mathcal{P}) = \cup_{v \in M} E(P_v)$ installed in the solution is given by $k_{\mathcal{P}}(e) = \lceil (\alpha + \beta \sum_{v : e \in E(P_v)} q(v))/\lambda \rceil$.
Goal: Minimize the total installed cost, that is,

$$\sum_{e \in E(\mathcal{P})} k_{\mathcal{P}}(e)w(e).$$

The following lemma follows directly from the definitions of GCND and GCTR.

Theorem 6.3. *Let $I' = (G, w, \lambda, \alpha, \beta, s, M, q)$ and $I = (G, w, \kappa, \lambda, \alpha, \beta, s, M, q)$ be two instances of* GCND *and* GCTR*, respectively. Then the optimal value of I' is a lower bound to the optimal value of I.*

Proof. Let $opt(I)$ and $opt(I')$ denote the optimal values of I and I', respectively. Consider an optimal solution $(\mathcal{M}^* = \{Z_1, \ldots, Z_\ell\}, \mathcal{T}^* = \{T_1, \ldots, T_\ell\})$ to GCTR instance I. For each $i = 1, 2, \ldots, \ell$ and $v \in Z_i$, let P_v be the path from v to s in T_i. We observe that $\mathcal{P} = \{P_v \mid v \in M\}$ is a feasible solution to GCND instance I'. Moreover, for $E(\mathcal{P}) = \cup_{v \in M} E(P_v)$ and $E(\mathcal{T}^*) = \cup_{T_i \in \mathcal{T}^*} E(T_i)$, it hold $E(\mathcal{P}) = E(\mathcal{T}^*)$ and $k_{\mathcal{P}}(e) \leq h_{\mathcal{T}^*}(e)$. Hence, it holds

$$opt(I') \leq \sum_{e \in E(\mathcal{P})} k_{\mathcal{P}}(e)w(e) \leq \sum_{e \in E(\mathcal{T}^*)} h_{\mathcal{T}^*}(e)w(e) = opt(I).$$

□

Before constructing an approximate solution to GCND, we present two lower bounds to the problem. The first lower bound is based on the Steiner tree problem, where the proof is similar to that of Lemma 6.1.

Lemma 6.5. *Given a* GCND *instance $I' = (G, w, \lambda, \alpha, \beta, s, M, q)$, the minimum cost of a Steiner tree that spans $M \cup \{s\}$ is a lower bound on the optimal value to I'.*

The second lower bound is based on a linear combination of both the Steiner tree problem and the distances from s to all terminals.

Lemma 6.6. *Let $I' = (G, w, \lambda, \alpha, \beta, s, M, q)$ be an instance of* GCND *and T^* be a minimum cost Steiner tree that spans $M \cup \{s\}$. Then*

$$(\alpha/\lambda) \sum_{e \in E(T^*)} w(e) + (\beta/\lambda) \sum_{v \in M} q(v) d_{(G,w)}(s, v)$$

is a lower bound on the optimal value to I'.

Proof. Consider an optimal solution $\mathcal{P} = \{P_v \mid v \in M\}$ to GCND instance I', and let $E(\mathcal{P}) = \cup_{v \in M} E(P_v)$. Let $opt(I')$ denote the optimal value to I'. Then we have

$$\begin{aligned} opt(I') &= \sum_{e \in E(\mathcal{P})} \left\lceil (\alpha + \beta \sum_{v : e \in E(P_v)} q(v))/\lambda \right\rceil w(e) \\ &\geq (\alpha/\lambda) \sum_{e \in E(\mathcal{P})} w(e) + (\beta/\lambda) \sum_{e \in E(\mathcal{P})} (w(e) \sum_{v : e \in E(P_v)} q(v)) \\ &= (\alpha/\lambda) \sum_{e \in E(\mathcal{P})} w(e) + (\beta/\lambda) \sum_{v \in M} (q(v) \sum_{e \in E(P_v)} w(e)) \\ &\geq (\alpha/\lambda) \sum_{e \in E(T^*)} w(e) + (\beta/\lambda) \sum_{v \in M} q(v) d_{(G,w)}(s, v), \end{aligned}$$

since $E(\mathcal{P})$ contains a tree that spans $M \cup \{s\}$ in G and $\sum_{e \in E(P_v)} w(e) \geq d_{(G,w)}(s, v)$ holds for all $v \in M$. □

Now we construct an approximate solution to a GCND instance $I' = (G, w, \lambda, \alpha, \beta, s, M, q)$ based on a tree balanced an approximate Steiner tree and a shortest path tree in G. Let T^* and T^{ast} denote optimal and ρ_{ST}-approximate solutions to the Steiner tree problem to $(G, w, M \cup \{s\})$, respectively. This implies that $w(T^{ast}) \leq \rho_{\mathrm{ST}} \cdot w(T^*)$. Regard T^* and T^{ast} as trees rooted at s. Let T^{spt} be a shortest path tree that spans $M \cup \{s\}$ rooted at s. Let T be a balanced Steiner tree that approximates both T^{ast} and T^{spt}. Note that T can be found in polynomial time (refer to Section 1.4 for details). Namely, given T^{ast}, T^{spt}, and a real number $\gamma > 0$, there is a balanced Steiner tree T such that

$$w(T) \leq (1 + 2/\gamma) w(T^{ast}), \text{ and} \tag{6.7}$$

$$d_{(T,w)}(s, v) \leq (1 + \gamma) d_{(G,w)}(s, v), \quad \text{for all } v \in M. \tag{6.8}$$

Let v^e denote the tail of edges e in T. Inequalities (6.7) and (6.8) imply that

$$\begin{aligned}
\sum_{e\in E(T)} \lceil(\alpha+\beta q(T_{v^e}))/\lambda\rceil w(e) &\leq \sum_{e\in E(T)} \big((\alpha+\beta q(T_{v^e}))/\lambda+1\big)w(e)\\
&= (\alpha/\lambda+1)w(T) + (\beta/\lambda)\sum_{v\in M} q(v)d_{(T,w)}(s,v)\\
&\leq (\alpha/\lambda+1)\rho_{\mathrm{ST}}(1+2/\gamma)w(T^*)\\
&\quad + (\beta/\lambda)(1+\gamma)\sum_{v\in M} q(v)d_{(G,w)}(s,v)\\
&\leq \rho_{\mathrm{ST}}(1+2/\gamma)w(T^*) + \max\{\rho_{\mathrm{ST}}(1+2/\gamma),(1+\gamma)\}\\
&\quad \big((\alpha/\lambda)w(T^*) + (\beta/\lambda)\sum_{v\in M} q(v)d_{(G,w)}(s,v)\big). \qquad (6.9)
\end{aligned}$$

Hence Lemmas 6.5 and 6.6 prove that the right hand side of (6.9) is bounded from above by

$$\big(\rho_{\mathrm{ST}}(1+2/\gamma) + \max\{\rho_{\mathrm{ST}}(1+2/\gamma),(1+\gamma)\}\big)opt(I'),$$

where $opt(I')$ denotes the optimal value to I'. This proves the following theorem.

Theorem 6.4. *Let $I' = (G, w, \lambda, \alpha, \beta, s, M, q)$ be an instance of* GCND *with optimal value $opt(I')$. Then, for any $\gamma > 0$, there is a Steiner tree T that spans $M \cup \{s\}$ rooted at s such that*

$$\sum_{e\in E(T)} \lceil(\alpha+\beta q(T_{v^e}))/\lambda\rceil w(e) \leq \mu \cdot opt(I'),$$

where v^e is the tail of e in T and $\mu = \rho_{\mathrm{ST}}(1+2/\gamma)+\max\{\rho_{\mathrm{ST}}(1+2/\gamma),(1+\gamma)\}$. Furthermore, such a tree T can be computed in polynomial time. □

6.4.2 Approximation algorithms to GCTR

In this section we present two approximation algorithms for a GCTR instance with $\lambda < \alpha+\beta\kappa$. Our proposed algorithms are based on κ-balanced partition and the results described in Section 6.4.1.

Algorithm ApproxGCTR
Input: An instance $I = (G, w, \kappa, \lambda, \alpha, \beta, s, M, q)$ of GCTR.
Output: A solution $(\mathcal{M}, \mathcal{T})$ to I.

Step 1. Compute a tree T that spans $M \cup \{s\}$ rooted at s.
Find a κ-balanced partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_p\}$ of M in T.

Step 2. For each $i = 1, 2, \ldots, p-1$, assign a vertex t_{Z_i} in $T\langle Z_i\rangle$ as its hub vertex and let T_{Z_i} be the tree obtained from $T\langle Z_i\rangle$ by adding the edge set of a shortest path $SP(s, t_{Z_i})$ between s and t_{Z_i} in G.
Let $t_{Z_p} := s$ and $T_{Z_p} := T\langle Z_p \cup \{s\}\rangle$.

Step 3. For each $i = 1, 2, \ldots, p$,
Regard T_{Z_i} as a tree rooted at s.
Install $\lceil(\alpha + \beta q(Z_i \cap D_{T_{Z_i}}(v_i^e)))/\lambda\rceil$ copies of each edge $e \in E(T_{Z_i})$ with tail v_i^e in T_{Z_i}.

Step 4. Let $\mathcal{T} = \{T_{Z_i} \mid i = 1, 2, \ldots, p\}$ and output $(\mathcal{M}, \mathcal{T})$.

Note that the demand capacity constraint on each tree in $\mathcal{T}$ is obviously satisfied by the definition of κ-balanced partition. It is also easy to observe that the edge capacity constraint remains satisfied on each edge installed on the graph. Thereby $(\mathcal{M}, \mathcal{T})$ is feasible to I. It remains to discuss the approximation ratio of the algorithm. We consider two versions of algorithm ApproxGCTR by realizing Steps 1 and 2 in two different ways as follows.

(A) We compute a tree T in the first step by any ρ_{ST}-approximation algorithm to the Steiner tree problem, and choose $t_{Z_i} \in Z_i$, $i = 1, 2, \ldots, p-1$, in Step 2 to be a terminal of the minimum distance $d_{(G,w)}(s, t_{Z_i})$ in Z_i, and

(B) we compute a tree T in the first step by using Theorem 6.4, and, for each $i = 1, 2, \ldots, p-1$, we choose t_{Z_i} in Step 2 to be a vertex of the minimum depth in T.

Theorem 6.5. *For a* GCTR *instance I with $\lambda < \alpha + \beta\kappa$, algorithm* ApproxGCTR *with Steps 1 and 2 as defined in* **(A)** *delivers an approximate solution $(\mathcal{M}, \mathcal{T})$ with approximation ratio of $2\xi + \min\{\lceil(\alpha + \beta\kappa)/\lambda\rceil, \lceil\beta\kappa/\lambda\rceil + 1\}\rho_{\mathrm{ST}}$, where $\xi = \lambda\lceil(\alpha + \beta\kappa)/\lambda\rceil/(\alpha + \beta\kappa)$.*

Proof. By construction and since $\alpha + \beta q(Z_i \cap D_{T_{Z_i}}(v_i^e)) \leq \alpha + \beta q(Z_i) \leq \alpha + \beta\kappa$, $i = 1, 2, \ldots, p$, the total cost of $(\mathcal{M}, \mathcal{T})$ is bounded from above by

$$\lceil(\alpha + \beta\kappa)/\lambda\rceil w(T) + \lceil(\alpha + \beta\kappa)/\lambda\rceil \sum_{1 \leq i \leq p-1} d_{(G,w)}(s, t_{Z_i}). \tag{6.10}$$

Let $opt(I)$ denote the optimal value of I. We first show that the second term in (6.10) is bounded by $2\xi opt(I)$, i.e.,

$$\sum_{1 \leq i \leq p-1} d_{(G,w)}(s, t_{Z_i}) \leq 2\lambda/(\alpha + \beta\kappa) opt(I). \tag{6.11}$$

Since $d_{(G,w)}(s,t) \geq d_{(G,w)}(s, t_{Z_i})$ for all $t \in Z_i$, $i = 1, 2, \ldots, p-1$, and $q(Z_i) > \kappa/2$ for all $i = 1, 2, \ldots, p-1$, we have

$$\begin{aligned} opt(I) &\geq (\alpha + \beta\kappa)/(\lambda\kappa) \sum_{t \in M} q(t) d_{(G,w)}(s,t) \qquad \text{(Lemma 6.2)} \\ &\geq (\alpha + \beta\kappa)/(\lambda\kappa) \sum_{1 \leq i \leq p-1} q(Z_i) d_{(G,w)}(s, t_{Z_i}) \\ &> (\alpha + \beta\kappa)/(2\lambda) \sum_{1 \leq i \leq p-1} d_{(G,w)}(s, t_{Z_i}). \end{aligned}$$

This completes the proof of (6.11).

Next we show that the first term of (6.10) is bounded by $\rho_{\rm ST}\lceil(\alpha+\beta\kappa)/\lambda\rceil opt(I)$ and $\rho_{\rm ST}(\lceil\beta\kappa/\lambda\rceil+1)opt(I)$.

For a minimum Steiner tree T^* that spans $M\cup\{s\}$, we have $w(T)\le\rho_{\rm ST}\cdot w(T^*)$ and $w(T^*)\le opt(I)$ by Lemma 6.1. Hence the first term of (6.10) is bounded by $\lceil(\alpha+\beta\kappa)/\lambda\rceil w(T)\le\rho_{\rm ST}\lceil(\alpha+\beta\kappa)/\lambda\rceil opt(I)$.

On the other hand, $\lceil\alpha/\lambda\rceil w(T^*)\le opt(I)$ by Lemma 6.4, and hence the first term of (6.10) is bounded by

$$\lceil(\alpha+\beta\kappa)/\lambda\rceil w(T) \;\le\; (\lceil\alpha/\lambda\rceil+\lceil\beta\kappa/\lambda\rceil)w(T)\le\rho_{\rm ST}(\lceil\beta\kappa/\lambda\rceil+1)opt(I).$$

This completes the proof of the theorem. □

Note that the ratio in Theorem 6.5 may not be constant due to the factor $\lceil\beta\kappa/\lambda\rceil$. We show in the next theorem that algorithm ApproxGCTR with Steps 1 and 2 as defined in **(B)** admits a constant factor approximate solution.

Theorem 6.6. *For a* GCTR *instance I with $\lambda<\alpha+\beta\kappa$, algorithm* ApproxGCTR *with Steps 1 and 2 as defined in* **(B)** *delivers an approximate solution $(\mathcal{M},\mathcal{T})$ with approximation ratio of $2\xi+2\rho_{\rm ST}+4\sqrt{2\xi\rho_{\rm ST}}$, where $\xi=\lambda\lceil(\alpha+\beta\kappa)/\lambda\rceil/(\alpha+\beta\kappa)$.*

Proof. Let e be an edge in $T\langle Z_i\rangle$ with tail v_i^e, $i=1,2,\ldots,p$. By property (iii) of κ-balanced partition and the choice of t_{Z_i}, we conclude that

$$\alpha+\beta q(Z_i\cap D_{T_{Z_i}}(v_i^e))\le\alpha+\beta q(T_{v_i^e}).$$

On the other hand, $\alpha+\beta q(Z_i\cap D_{T_{Z_i}}(v_i^e))\le\alpha+\beta q(Z_i)\le\alpha+\beta\kappa$ holds for $i=1,2,\ldots,p$. Hence the total weight of the installed edges on the network is bounded by

$$\sum_{e\in E(T)}\lceil(\alpha+\beta q(T_{v^e}))/\lambda\rceil w(e)+\lceil(\alpha+\beta\kappa)/\lambda\rceil\sum_{1\le i\le p-1}d_{(G,w)}(s,t_{Z_i}),\tag{6.12}$$

where v^e is the tail of e in T.

Let $opt(I)$ denote the optimal value to I. For a Steiner tree T computed in Step 1, Theorems 6.3 and 6.4 imply that

$$\sum_{e\in E(T)}\lceil(\alpha+\beta q(T_{v^e}))/\lambda\rceil w(e)\le[\rho_{\rm ST}(1+2/\gamma)+\max\{\rho_{\rm ST}(1+2/\gamma),(1+\gamma)\}]opt(I).\tag{6.13}$$

On the other hand, by the choice of t_{Z_i}, $i=1,2,\ldots,p-1$, we have $d_{(T,w)}(s,t_{Z_i})\le d_{(T,w)}(s,t)$ for all $t\in Z_i$, and hence it holds

$$d_{(G,w)}(s,t_{Z_i})\le d_{(T,w)}(s,t_{Z_i})\le d_{(T,w)}(s,t)\le(1+\gamma)d_{(G,w)}(s,t),\ \text{for all } t\in Z_i.$$

From this and $q(Z_i) > \kappa/2$ for all $i = 1, 2, \ldots, p-1$, we have

$$\begin{aligned} opt(I) &\geq (\alpha+\beta\kappa)/(\lambda\kappa)\sum_{t\in M} q(t)d_{(G,w)}(s,t) \qquad \text{(Lemma 6.2)} \\ &\geq (\alpha+\beta\kappa)/(\lambda\kappa(1+\gamma))\sum_{1\leq i\leq p-1} q(Z_i)d_{(G,w)}(s,t_{Z_i}) \\ &> (\alpha+\beta\kappa)/(2\lambda(1+\gamma))\sum_{1\leq i\leq p-1} d_{(G,w)}(s,t_{Z_i}). \qquad (6.14) \end{aligned}$$

Now, by using (6.13) and (6.14), we conclude that (6.12) is at most

$$[2\xi(1+\gamma)+\rho_{\mathrm{ST}}(1+2/\gamma)+\max\{\rho_{\mathrm{ST}}(1+2/\gamma),(1+\gamma)\}]opt(I),$$

where $\xi = \lambda\lceil(\alpha+\beta\kappa)/\lambda\rceil/(\alpha+\beta\kappa)$. Note that $\xi \in [1,2)$. Such a factor is minimized by choosing $\gamma = \sqrt{2\rho_{\mathrm{ST}}/\xi}$. This implies that the total weight of the installed edges is bounded from above by

$$(2\xi + 2\rho_{\mathrm{ST}} + 4\sqrt{2\xi\rho_{\mathrm{ST}}}\,)opt(I).$$

□

Note that the approximation ratio given in Theorem 6.6 is bounded from above by

$$(2\xi + 2\rho_{\mathrm{ST}} + 4\sqrt{2\mu\rho_{\mathrm{ST}}}\,) < (4 + 2\rho_{\mathrm{ST}} + 8\sqrt{\rho_{\mathrm{ST}}}\,) < 17.057$$

for the best known ratio $\rho_{\mathrm{ST}} = 1 + \frac{\ln 3}{2}$ to the Steiner tree problem (since $\xi < 2$).

We show that the bound can be improved by choosing the best one from both solutions constructed by using **(A)** and **(B)** in Steps 1 and 2.

Theorem 6.7. *For a* GCTR *instance I with $\lambda < \alpha+\beta\kappa$, there exists an approximate solution $(\mathcal{M}, \mathcal{T})$ with approximation ratio of*

$$\min\{2\xi + \lceil(\alpha+\beta\kappa)/\lambda\rceil\rho_{\mathrm{ST}}, 2\xi + 2\rho_{\mathrm{ST}} + 4\sqrt{2\xi\rho_{\mathrm{ST}}}\} \leq 13.037.$$

Proof. Let $j = \lceil(\alpha+\beta\kappa)/\lambda\rceil$. Note that $\lambda < \alpha+\beta\kappa$ implies that $j = \lceil(\alpha+\beta\kappa)/\lambda\rceil \geq 2$. Since $j-1 < (\alpha+\beta\kappa)/\lambda \leq j$, ξ is bounded from above by

$$\xi = \lambda\lceil(\alpha+\beta\kappa)/\lambda\rceil/(\alpha+\beta\kappa) < j/(j-1).$$

First consider the case where $\lceil(\alpha+\beta\kappa)/\lambda\rceil \leq 6$. In this case, for the best known ratio $\rho_{\mathrm{ST}} = 1 + \frac{\ln 3}{2}$ to the Steiner tree problem, the approximation factor $2\xi + \lceil(\alpha+\beta\kappa)/\lambda\rceil\rho_{\mathrm{ST}}$ proved in Theorem 6.5 is bounded from above by

$$2\xi + \lceil(\alpha+\beta\kappa)/\lambda\rceil\rho_{\mathrm{ST}} \leq 11.696,$$

which is obtained when $j = \lceil(\alpha+\beta\kappa)/\lambda\rceil = 6$ (and hence $\xi < j/(j-1) = 6/5$).

Next consider the case where $\lceil(\alpha+\beta\kappa)/\lambda\rceil \geq 7$. We have $\xi < j/(j-1) \leq 7/6$ and hence the approximation factor $2\xi + 2\rho_{\mathrm{ST}} + 4\sqrt{2\xi\rho_{\mathrm{ST}}}$ proved in Theorem 6.6 is bounded from above by

$$2\xi + 2\rho_{\mathrm{ST}} + 4\sqrt{2\xi\rho_{\mathrm{ST}}} \leq 13.037$$

since $2\xi + 2\rho_{\mathrm{ST}} + 4\sqrt{2\xi\rho_{\mathrm{ST}}}$ is an increasing function of ξ over $[1, 2)$. This completes the proof of the theorem. □

6.5 Approximation algorithm to FGCTR

In this section we present an approximation algorithm for a FGCTR instance by modifying the algorithm given in Section 6.4.2. We first introduce the following lower bound on the optimal value to FGCTR. The proof of the lemma is similar to that of Lemma 6.2.

Lemma 6.7. *Let $I = (G, w, \kappa, \alpha, \beta, s, M, q)$ be an instance of* FGCTR*. Then*

$$(\alpha+\beta\kappa)/\kappa \sum_{v\in M} q(v) d_{(G,w)}(s, v)$$

is a lower bound on the optimal value to I. □

The fractional generalized capacitated network design problem (FGCND) is a variant of GCND in which it is allowed to purchase edge capacity in any required quantity. Namely, we assign capacity of $\lambda_e = \alpha + \beta \sum_{v:e\in E(P_v)} q(v)$ on each edge e in $E(\mathcal{P}) = \cup_{v\in M} E(P_v)$. That is, the total cost of installed capacities equals $\sum_{e\in E(\mathcal{P})} \lambda_e w(e)$. Corresponding results to that in Sections 6.4.1 and 6.4.2 can be obtained similarly.

Theorem 6.8. *Let $I' = (G, w, \alpha, \beta, s, M, q)$ and $I = (G, w, \kappa, \alpha, \beta, s, M, q)$ be two instances of* FGCND *and* FGCTR*, respectively. Then the optimal value to I' is a lower bound on the optimal value to I.* □

Theorem 6.9. *Let $I' = (G, w, \alpha, \beta, s, M, q)$ be an instance of* FGCND *and let $opt(I')$ be the optimal value to I'. Then, for any $\gamma > 0$, there is a Steiner tree T that spans $M \cup \{s\}$ rooted at s such that*

$$\sum_{e\in E(T)} (\alpha + \beta q(T_{v^e}))w(e) \leq \max\{\rho_{\mathrm{ST}}(1+2/\gamma), (1+\gamma)\} opt(I'),$$

where v^e is the tail of e in T. □

Now, we are ready to present a formal algorithm to FGCTR based on the above results.

Algorithm ApproxFGCTR
Input: An instance $I = (G, w, \kappa, \alpha, \beta, s, M, q)$ of FGCTR.
Output: A solution $(\mathcal{M}, \mathcal{T})$ to I.

Step 1. Compute a $(\max\{\rho_{\mathrm{ST}}(1+2/\gamma),(1+\gamma)\})$-approximate Steiner tree T that spans $M \cup \{s\}$ rooted at s by Theorem 6.9.
Find a κ-balanced partition $\mathcal{M} = \{Z_1, Z_2, \ldots, Z_p\}$ of M in T.

Step 2. For each $i = 1, 2, \ldots, p-1$, choose a vertex t_{Z_i} in $T\langle Z_i\rangle$ with the minimum depth in T and let T_{Z_i} be the tree obtained from $T\langle Z_i\rangle$ by adding the edge set of a shortest path $SP(s, t_{Z_i})$ between s and t_{Z_i} in G.
Let $t_{Z_p} := s$ and $T_{Z_p} := T\langle Z_p \cup \{s\}\rangle$.

Step 3. Let $\mathcal{T} = \{T_{Z_i} \mid i = 1, 2, \ldots, p\}$ and output $(\mathcal{M}, \mathcal{T})$.

Theorem 6.10. *For a* FGCTR *instance I, algorithm* ApproxFGCTR *delivers an approximate solution $(\mathcal{M}, \mathcal{T})$ with approximation ratio of* 8.529.

Proof. By construction, the total cost of $(\mathcal{M}, \mathcal{T})$ is bounded from above by

$$\sum_{e \in E(T)} (\alpha + \beta q(T_{v^e}))w(e) + (\alpha + \beta\kappa) \sum_{1 \le i \le p-1} d_{(G,w)}(s, t_{Z_i}), \tag{6.15}$$

where v^e is the tail of e in T. Let $opt(I)$ denote the optimal value to I. For a Steiner tree T computed in Step 1, Theorems 6.9 and 6.8 imply that

$$\sum_{e \in E(T)} (\alpha + \beta q(T_{v^e}))w(e) \le \max\{\rho_{\mathrm{ST}}(1+2/\gamma),(1+\gamma)\}opt(I). \tag{6.16}$$

On the other hand, for each $i = 1, 2, \ldots, p-1$, the choice of t_{Z_i} implies that

$$d_{(G,w)}(s, t_{Z_i}) \le d_{(T,w)}(s, t_{Z_i}) \le d_{(T,w)}(s, t) \le (1+\gamma) d_{(G,w)}(s, t), \text{ for all } t \in Z_i.$$

From this and $q(Z_i) > \kappa/2$ for all $i = 1, 2, \ldots, p-1$, we have

$$\begin{aligned} opt(I) &\ge (\alpha + \beta\kappa)/\kappa \sum_{t \in M} q(t) d_{(G,w)}(s, t) \qquad \text{(Lemma 6.7)} \\ &\ge (\alpha + \beta\kappa)/(\kappa(1+\gamma)) \sum_{1 \le i \le p-1} q(Z_i) d_{(G,w)}(s, t_{Z_i}) \\ &> (\alpha + \beta\kappa)/(2(1+\gamma)) \sum_{1 \le i \le p-1} d_{(G,w)}(s, t_{Z_i}), \end{aligned} \tag{6.17}$$

Now, by using (6.16) and (6.17), we conclude that (6.15) is at most

$$[2(1+\gamma) + \max\{\rho_{\mathrm{ST}}(1+2/\gamma),(1+\gamma)\}]opt(I),$$

which is minimized by choosing $\gamma = \sqrt{\rho_{\mathrm{ST}}}$. This implies that, for the best known ratio $\rho_{\mathrm{ST}} = 1 + \frac{\ln 3}{2}$ to the Steiner tree problem, the total cost of $(\mathcal{M}, \mathcal{T})$ is bounded from above by $(2 + \rho_{\mathrm{ST}} + 4\sqrt{\rho_{\mathrm{ST}}})opt(I) \le 8.529 opt(I)$, which proves the theorem. □

Chapter 7

Conclusion

In this thesis, we present approximation algorithms of several capacitated tree-routing problems in networks. The results obtained in the thesis are summarized as follows.

In Chapter 2, we study the capacitated multicast tree routing problem (CMTR) in networks. For CMTR instances with a general demand function, we have designed a $(2+\rho_{\rm ST})$-approximation algorithm, where $\rho_{\rm ST}$ is any approximation ratios achievable for the Steiner tree problem. The best known approximation ratio of the Steiner tree problem is $1+\frac{\ln 3}{2}<1.55$ for general graphs [62].

Next, we have designed a $(3/2+(4/3)\rho_{\rm ST})$-approximation algorithm for the unit demand case of CMTR. Our algorithm outperforms the $(3/2+(7/5)\rho_{\rm ST})$-approximation algorithm which is designed for the L_p metric in the plane [40]. Our approximation ratio also improves that obtained by Cai et al. [11] in the case of $\rho_{\rm ST}<1.2$. In particular, it is known that $\rho_{\rm ST}=1$ when $M=V$ since the Steiner tree problem with terminal set $M=V$ in G becomes the minimum spanning tree problem. Hence our approximation ratio improves that of Cai et al. [11] in the case where $M=V$. It is left as a future work to obtain a better approximation algorithm than $(8/5+(5/4)\rho_{\rm ST})$-approximation algorithm due to Cai et al. [11] in the case of $1.2<\rho_{\rm ST}<1.55$.

In Chapter 3, we have presented a $(2\rho_{\rm UFL}+\rho_{\rm ST})$-approximation algorithm for the capacitated multi-source multicast tree routing problem (CMMTR) with a general demand function, where $\rho_{\rm UFL}$ is any approximation ratio achievable for the metric UFL problem. Since the current best approximation ratios for UFL and the Steiner tree problems are 1.52 [52] and 1.55 [62], respectively, our algorithm delivers a 4.59-approximate solution to the problem. When all terminals of CMMTR have unit demands, we have used the result on the tree cover problem described in Chapter 2 to design an algorithm with a better approximation ratio $(3/2)\rho_{\rm UFL}+(4/3)\rho_{\rm ST}$, which is 4.35 in term of the current best approximation ratios for the metric UFL and the Steiner tree problems. Both of these algorithms are based on lower

bounds based on the Steiner tree and the metric UFL problems. Any improvement over approximation to the Steiner tree or UFL problem will be reflected in the approximation ratios of our algorithms. On the other hand, the coefficients of ρ_{UFL} and ρ_{ST} in the approximation ratios are induced from the tree cover results. Hence it would be interesting to get a better result on tree covers in order to improve the current approximation ratios to CMMTR.

In Chapter 4, we have studied the minimum cost edge installation problem (MCEI), a problem of finding a routing from a set of sources to a single sink in a network with an edge installing cost. MCEI is closely related to the capacitated network design problem (CND). In particular, a solution to each of MCEI and CND can be characterized by a set of paths, each of which sends the demand of a source to the sink and the set of these paths induces the numbers of cables installed on each edge of the network. CND allows the demand from a source to be split into fractions which pass through different copies of the same edge, while MCEI does not allow such splitting. The algorithm of Hassin et al. [33] to CND can be applied to MCEI instances to obtain approximate solutions of approximation ratios of $1+\rho_{\text{ST}}$ and $2+\rho_{\text{ST}}$ for the unit and general demand networks, respectively. We have designed a $(15/8+\rho_{\text{ST}})$-approximation algorithm for MCEI with general demand, improving that of Hassin et al. [33].

As a future work, we discuss a possible generalization of MCEI and CND, in which we concerned with multiple sinks. In a general problem setting for routing problems, a group of vertices of the underlying network is designated as sinks such that each sink is associated with an opening cost. In this case, the problem asks to open a set of sinks and construct a set of paths, each of which sends the demand of a source to an opened sink, minimizing the cost of installed edges and opened sinks. As mentioned in Chapter 3, Ravi and Sinha [61] already studied such a multi-sink version of CND, called it CCFL, and gave a $(\rho_{\text{UFL}}+\rho_{\text{ST}})$-approximation algorithm for CCFL with the unit demands and a $(2\rho_{\text{UFL}}+\rho_{\text{ST}})$-approximation algorithm for that with the general demands. It would be interesting to investigate approximation algorithms for such a multi-sink version of MCEI.

In Chapter 5, we have studied the capacitated tree-routing problem (CTR), a new routing problem formulation under a multi-tree model which unifies several important routing problems such as CMTR and the unit demand case of CND. We have proved that CTR is $(2+\rho_{\text{ST}})$-approximable based on new results on tree covers. It would be interesting to investigate a CTR version of MCEI and the general demand case of CND.

In Chapter 6, we have studied a more general routing model, the generalized capacitated tree-routing problem (GCTR), a new routing problem formulation under a multi-tree model with a general routing capacity, which unifies several important routing problems such as CND, CMTR, and CTR. We have proved that GCTR with $\lambda \geq \alpha+\beta\kappa$ is

$([2\lambda/(\alpha+\beta\kappa)]/\lfloor\lambda/(\alpha+\beta\kappa)\rfloor+\rho_{\mathrm{ST}})$-approximable based on the algorithm used in Chapter 5. Note that, in this case, $[\lambda/(\alpha+\beta\kappa)]/\lfloor\lambda/(\alpha+\beta\kappa)\rfloor < 2$ holds. For $\lambda < \alpha+\beta\kappa$, we introduced a new lower bound on GCTR by formulating a generalization of CND, and use a balanced Steiner tree as a base tree from which we construct a collection of trees to send demands to sink. We show that our new algorithm delivers a 13.037-approximate solution to GCTR with $\lambda < \alpha+\beta\kappa$.

We also have studied a natural variant of GCTR, the fractional generalized capacitated tree-routing problem (FGCTR), in which it is allowed to purchase edge capacity in any required quantity. We have designed an approximation algorithm to FGCTR with approximation ratio of at most 8.529.

Future work may include design of approximation algorithms for further extensions of our tree-routing model. One possible extension is to extend GCTR and FGCTR to multi-sink versions.

Bibliography

[1] M. Andrews and L. Zhang, The access network design problem, In Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS) (1998) 40-49.

[2] S. Arora, The Approximability of NP-Hard Problems, In Proceedings of the 30th ACM Symposium on Theory of Computing (STOC) (1998) 337-348.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, Springer (1999).

[4] B. Awerbuch and Y. Azar, Buy-at-bulk network design, In Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS) (1997) 542-547.

[5] B. Awerbuch, A. Baratz, and D. Peleg, Efficient broadcast and light-weight spanners, Technical Report CS92-22, Weizmann Institute of Science (1992).

[6] L. Behjat, New modeling and optimization techniques for the global routing problem, Ph.D. Thesis, University of Waterloo (2002).

[7] L. Behjat, A. Vannelli, and W. Rosehart, Integer linear programming models for global routing, Informs Journal on Computing, 18(2) (2002) 137-150.

[8] P. Berman and V. Ramaiyer, Improved approximations for the Steiner tree problem, J. Algorithms, 17 (1994) 381-408

[9] K. Bharath-Kumar, and J. M. Jaffe, Routing to multiple destinations in computer networks, IEEE Transactions on Communications, 31 (1983) 343-351.

[10] Z. Cai, G.-H. Lin, and G. Xue, Improved approximation algorithms for the capacitated multicast routing problem, In Proceedings of the 11th Annual International Computing and Combinatorics Conference (COCOON), LNCS 3595 (2005) 136-145.

[11] Z. Cai, Z.-Z. Chen, G.-H. Lin, L. Wang, An improved approximation algorithm for the capacitated multicast tree routing problem, In Proceedings of the 2nd Annual International Conference on Combinatorial Optimization and Applications (COCOA), LNCS 5165 (2008) 286-295.

[12] M. Charikar and S. Guha, Improved combinatorial algorithms for facility location and k-median problems, In Proceedings of the 40th Annual IEEE Symposium on Foundation of Computer Science (FOCS) (1999) 378-388.

[13] F. A. Chudak, Improved approximation algorithms for uncapacitated facility location, In Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization (IPCO), LNCS 1412 (1998) 180-194.

[14] S. A. Cook, The complexity of theorem-proving procedures, In Proceedings of the 3th Annual ACM Symposium on Theory of Computing (STOC) (1971) 151-158.

[15] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey, The uncapacitated facility location problem, Discrete Location Theory Wiley-Interscience, New York (1990) 119-171.

[16] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, Performance-driven global routing for cell based ICs, In Proceedings of IEEE International Conference on Computer Design (ICCD) (1991) 170-173.

[17] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, Provably good performance-driven global routing, IEEE Transactions on CAD (1992) 739-752.

[18] P. Crescenzi and V. Kann, A compendium of NP optimization problems, Technical Report, Dipartimento di Scienz dell'Informazione, Universita di Roma (1995) http://www.nada.kth.se/~viggo/problemlist/compendium.html.

[19] S. Dreyfus and R. Wagner, The Steiner problem in graphs, Networks, 1 (1972) 195-207.

[20] N. Garg, R. Khandekar, G. Konjevod, R. Ravi, F.S. Salman, and A. Sinha, On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design problem, In Proceedings of the 8th International Conference on Integer Programming and Combinatorial Optimization (IPCO), LNCS 2081 (2001) 170-184.

[21] M. R. Garey and D. S. Johnson, The rectilinear Steiner tree problem is NP-complete, SIAM J. Appl. Math., 32 (1977) 826-843.

[22] M. R. Garey and D. S. Johnson, Computers and Inntractability: A Guide to the Theory of NP-Completeness, W. H. Freeman (1979).

[23] E. N. Gilbert and H. O. Pollak, Steiner minimal trees, SIAM J. Appl. Math., 16 (1968) 1-29.

[24] F. Grandoni and G. F. Italiano, Improved approximation for single-sink buy-at-bulk, In Proceedings of International Symposium on Algorithms and Computation (ISAAC), LNCS 4288 (2006) 111-120.

[25] J. Gu, X.-D. Hu, and M.-H. Zhang, Algorithm for multicast connection under multi-path routing Model, Information Processing Letters, 84 (2002) 31-39.

[26] J. Gu, X.-D. Hu, and M.-H. Zhang, Routing algorithm for multicast under multi-tree model in optical networks, Theoretical Computer Science, 314 (2004) 293-301.

[27] S. Guha and S. Khuller, Greedy strikes back: Improved facility location algorithms, Journal of Algorithms, 31 (1999) 228-248.

[28] S. Guha, A. Meyerson, and K. Munagala, Hierarchical placement and network design problems, In Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science (FOCS) (2000) 603-612.

[29] S. Guha, A. Meyerson, and K. Munagala, Improved combinatorial algorithms for single sink edge installation problems, Technical Report STAN-CS-TN00-96, Stanford University (2000).

[30] S. Guha, A. Meyerson, and K. Munagala, A constant factor approximation for the single sink edge installation problem, In Proceedings of th 33th ACM symposium on the Theory of Computing (STOC) (2001) 383-388.

[31] A. Gupta, A. Kumar, and T. Roughgarden, Simpler and better approximation algorithms for network design, In Proceedings of the 35th ACM Symposium on Theory of Computing (STOC) (2003) 365-372.

[32] R. L. Hadas, Efficient collective communication in WDM networks with a power budget, In proceeding of the 9th IEEE International Conference on Computer Communications and Networks (2000) 612-616.

[33] R. Hassin, R. Ravi, and F. S. Salman, Approximation algorithms for a capacitated network design problem, Algorithmica, 38 (2004) 417-431.

[34] D. S. Hochbaum, editor, Approximation Algorithms for NP-hard Problems, PWS Publishing Company (1997).

[35] S. Hougardy and H. J. Prömmel, A 1.598 approximation algorithm for the Steiner problem in graphs, In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (1999) 448-453.

[36] C. Huitema, Routing in the internet, Prentice Hall PTR, Englewood Cliffs (2000).

[37] K. Jain, M. Mahdian, and A. Saberi, A new greedy approach for facility location problems, In proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC) (2002) 731-740.

[38] K. Jain and V. V. Vazirani, Approximation algorithms for the metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation, Journal of ACM, 48 (2001) 274-296.

[39] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, On the placement of internet instrumentations, In Proceedings of IEEE INFOCOM (2000) 26-30.

[40] R. Jothi and B. Raghavachari, Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design, In Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP), LNCS 3142 (2004) 805-818.

[41] R. Jothi and B. Raghavachari, Improved Approximation Algorithms for the Single-Sink Buy-at-Bulk Network Design Problems, In Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT), LNCS 3111 (2004) 336-348.

[42] R. M. Karp, Reducibility among combinatorial problems, In Proceedings of a Symposium on the Complexity of Computer Computations (1972) 85-103.

[43] M. Karpinsky and A. Zelikovsky, New approximation algorithms for the Steiner tree problem, J. Combin. Optim., 1 (1997) 47-65.

[44] S. Khuller, B. Raghavachari, and N. N. Young, Balancing minimum spanning and shortest path trees, Algorithmica, 14 (1993) 305-322.

[45] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, Analysis of a local search heuristic for facility location problems, In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (1998) 1-10.

[46] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, In Proceedings of Amer. Math. Soc., 7 (1956) 48-50.

[47] A. A. Kuehn and M.J. Hamburger, A heuristic program for locating warehouses, Management Science, 9 (1963) 643-666.

[48] F. Kuo, W. Effelsberg, and J. J. Garcia-Luna-Aceves, Multimedia communications: protocols and applications, Prentice Hall, Inc., Englewood Cliffs (1998).

[49] G.-H. Lin, An improved approximation algorithm for multicast k-tree routing, Journal of Combinatorial Optimization, 9 (2005) 349-356.

[50] G.-H. Lin and G. Xue, Balancing Steiner minimum trees and shortest-path trees in the rectilinear plane, In Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS), 6 (1999) 117-120.

[51] M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani, A greedy facility location algorithm analyzed using dual fitting, In Proceedings of the 5th International Workshop on Randomization and Approximation Techniques in Computer Science, LNCS 2129 (2001) 127-137.

[52] M. Mahdian, Y. Ye, and J. Zhang, A 1.52 approximation algorithm for the uncapacitated facility location problem, In Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), LNCS 2462 (2002) 229-242.

[53] Y. Mansour and D. Peleg, An approximation algorithm for minimum-cost network design, Tech. Report Cs94-22, The Weizman Institute of Science, Rehovot (1994); also presented at the DIMACS Workshop on Robust Communication Network (1998).

[54] A. Meyerson, K. Munagala, and S. Plotkin, Cost distance: two metric network design, In Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science (FOCS) (2000) 624-630.

[55] E. Morsy and H. Nagamochi, Approximation algorithms for multicast routings in a network with multi-sources, IEICE Transactions E90-A, 5 (2007) 900-906.

[56] E. Morsy and H. Nagamochi, An improved approximation algorithm for capacitated multicast routings in networks, Theoritical Computer Science, 390(1) (2008) 81-91.

[57] E. Morsy and H. Nagamochi, Approximating the generalized capacitated tree-routing problem, In Proceedings of the 14th Annual International Computing and Combinatorics Conference (COCOON), LNCS 5092 (2008) 621-630.

[58] R. C. Prim, Shortest connection networks and some generalizations, Bell System Technical Journal, 36 (1957) 1389-1401.

[59] H. J. Prömmel and A. Steger, A new approximation algorithm for the Steiner tree problem with performance ratio 5/3, J. Algorithms, 36 (2000) 89-101.

[60] L. Qiu, V. N. Padmanabhan, and G. Voelker, On the placement of web server replicas, In Proceedings of IEEE INFOCOM (2001) 1587-1596.

[61] R. Ravi and A. Sinha, Approximation algorithms for problems combining facility location and network, Operations Research, 54(1) (2006) 73-81.

[62] G. Robins and A. Z. Zelikovsky, Improved Steiner tree approximation in graphs, In Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (2000) 770-779.

[63] L. H. Sahasrabuddhe and B. Mukherjee, Light-trees: optical multicasting for improved performance in wavelength-routed networks, IEEE Comm. Mag., 37(2) (1999) 67-73.

[64] L. H. Sahasrabuddhe and B. Mukherjee, Multicast routing algorithms and protocols: a tutorial, IEEE Network, 14(1) (2000) 90-102.

[65] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian, Approximating the single-sink link-installation problem in network design, SIAM J. Optim., 11 (2000) 595-610.

[66] D. B. Shmoys, E. Tardos, and K. I. Aardal, Approximation algorithms for facility location problems, In Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC) (1997) 265-274.

[67] M. Sviridenko, An 1.582-approximation algorithm for the metric uncapacitated facility location problem, In Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO), LNCS 2337 (2002) 240-257.

[68] T. Talwar, Single-sink buy-at-bulk LP has constant integrality gap, In Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO), LNCS 2337 (2002) 475-486.

[69] T. Terlaky, A. Vannelli, and H. Zhang, On routing in VLSI design and communication networks, In Proceedings of International Symposium on Algorithms and Computation (ISAAC), LNCS 3827 (2005) 1051-1060.

[70] Z. Wang and J. Crowcroft, Quality-of-service routing for supporting multimedia applications, IEEE Journal on Selected Areas in Communications, 14 (1996) 1228-1234.

[71] A. Zelikovsky, An 11/6-approximation algorithm for the network Steiner problem, Algorithmica, 9 (1993) 463-470.

[72] A. Zelikovsky, Better approximation bounds for the network and Euclidean Steiner tree problems, Technical Report CS-96-06, University of Virginia (1996).

[73] L. Zhao and H. Yamamoto, Multisource receiver-driven layered multicast, In Proceedings of IEEE TENCON (2005) 1325-1328.

[74] H.-Z. Zheng, D.-H. Chu, and D.-C. Zhan, Effective algorithm CFL based on Steiner tree and UFL problem, IJCSNS, 6(9A) (2006) 24-27.

List of publications

Journals

1. E. Morsy and H. Nagamochi, Approximation algorithms for multicast routings in a network with multi-sources, IEICE Transactions vol. E90-A, no. 5, pp. 900-906, 2007.

2. E. Morsy and H. Nagamochi, An improved approximation algorithm for capacitated multicast routings in networks, Theoretical Computer Science, vol. 390(1): 81-91, 2008. (journal version of conference paper 1)

3. E. Morsy and H. Nagamochi, Approximating capacitated tree-routings in networks, Journal of Combinatorial Optimization, to appear. (journal version of conference paper 2)

4. E. Morsy and H. Nagamochi, On the approximation of the generalized capacitated tree-routing problem, Journal of Discrete Algorithms, to appear.

5. E. Morsy and H. Nagamochi, Approximation to the minimum cost edge installation problem (submitted). (journal version of conference paper 3)

Conferences

1. E. Morsy and H. Nagamochi, An improved approximation algorithm for capacitated multicast routings in networks, Proceedings of the International Symposium on Scheduling 2006 (ISS2006) July 18-20, 2006 in Arcadia Ichigaya, Tokyo, Japan, pp. 12-17.

2. E. Morsy and H. Nagamochi, Approximating capacitated tree-routings in networks, The 4th Annual Conference on Theory and Applications of Models of Computation (TAMC07) Shanghai, China, May 22 to 25, 2007, Lecture Notes in Computer Science, volume 4484, Springer 2007, 342-353.

3. E. Morsy and H. Nagamochi, Approximation to the minimum cost edge installation problem, The 18th International Symposium on Algorithms and Computation (ISAAC) December 17-19, 2007 Sendai, Japan, Lecture Notes in Computer Science, volume 4835, Springer 2007, 292-303.

4. E. Morsy and H. Nagamochi, Approximating the generalized capacitated tree-routing, The 14th Annual International Computing and Combinatorics Conference (COCOON 2008) June 27 -29, 2008, Dalian, China, Lecture Notes in Computer Science, volume 5092, Springer 2008, 621-630.